PROJECT 4
LOGISTIC REGRESSION AND ADALINE NETWORKS

DAVID ATLAS

ABSTRACT. This paper will introduce two parametric algorithms for classifying linearly seperable classes. Logistic Regression applies a log-odds transformation to the target class to make predictions on the likelihood of a instance belonging to a class. An Adaline Network attempts to predict the class value (0 or 1) using a linear transformation. Both algorithms are trained using gradient descent. This paper will also apply both classifiers to 5 real world datasets. It is expected that the algorithms will perform similarly on real world data, as they are both able to find linear discriminants, and thus have similar model capacity. Real-world experiments indicate that performance tends to be similar between the two algorithms, and both outperform baselines on the datasets used, indicating classes that are at least partially linearly seperable.

1. Problem Statement & Hypothesis

This paper introduces techniques to find a linear discriminant between classes, based on a set of attributes about the classes. The parameters of the linear discriminant are learned via a gradient descent type algorithm. Because both techniques are linear, it is expected that they perform comparably on any given problem, as neither has an advantage in capacity.

It is expected that both techniques will not be able to learn classifications on non-linearly seperable problems without first transforming features. In Figure 1, such a problem is shown.

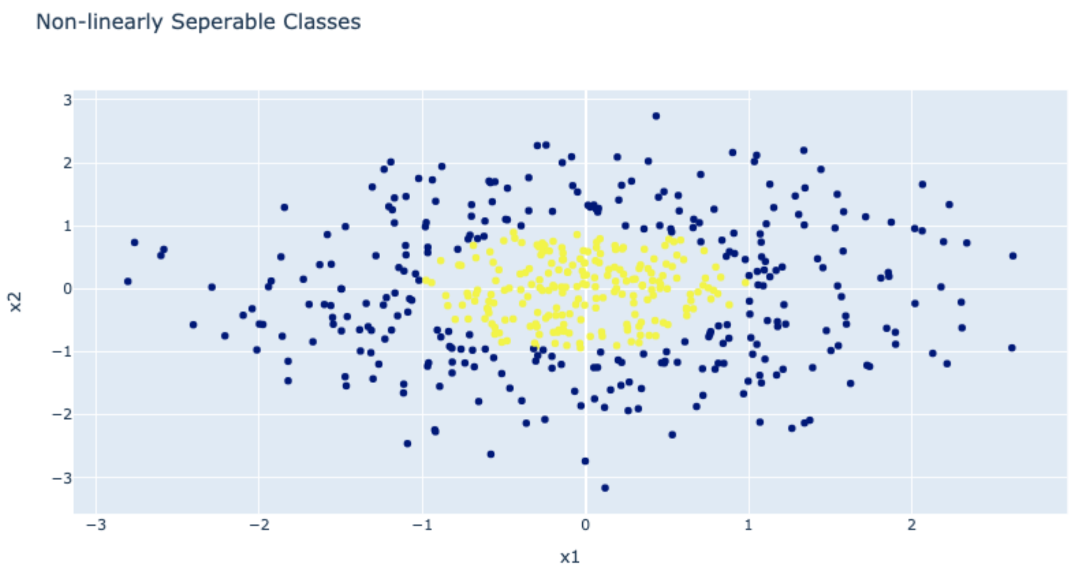

FIGURE 1. In the figure above, the yellow class cannot be seperated from the blue class via a single linear discriminant. In such a situation, the linear discriminant methods would be expected to perform poorly on a classification task.

As such, the classification accuracy of the algorithms will be inline with the proportion of instances that are linearly seperable. Problems that are mostly linearly seperable, but where labels are noisy, will have accuracy proportional to the degree of noise.

Because the algorithms both involve linear transformations of the features, they do not natively handle any discrete-multivalued attributes. Such inputs must be transformed to one-hot encodings. To assist with the convergence of the training algorithm, all inputs are rescaled such that all values are between 0 and 1.

The algorithms do not have capacity to identify interactions between input features. All such interactions must be identified manually and encoded as a seperate input. Likewise, non-linear relationships between features and the class will not be learned by the algorithm.

Both techniques are relatively efficient to train, although the gradient descent algorithm may not converge if problems are not linearly seperable, and the learning rate of the algorithm has a significant impact on convergence.

2. Description of Algorithm

2.1. **Logistic Regression.**

Discriminant Development. The logistic regression algorithm[3] creates a linear discriminant between two classes. Suppose k classes $(C_1, \cdots, C_k)$ exist. The logistic regression classifier seeks to calculate

$$\text{logit } P(C_k \mid x) = \log \frac{P(C_1 \mid x)}{1 - P(C_1 \mid x)}, \tag{1}$$

or the probability that instance x belongs to class C_k given the attributes of instance x.

Given Equation 1 and applying Bayes' Rule, it follows that

$$\log \frac{P(C_k \mid x)}{1 - P(C_k \mid x)} = \log \frac{P(x \mid C_k)}{1 - P(x \mid C_k)} + \log \frac{P(C_k)}{1 - P(C_k)}.$$

This will be expressed as a linear discriminant:

$$\log \frac{P(x \mid C_k)}{1 - P(x \mid C_k)} + \log \frac{P(C_k)}{1 - P(C_k)} = w_k^T x + w_{k_0}. \tag{2}$$

Using a bit of algebra, Equation 2 can be turned into

$$P(C_k \mid x) = \frac{\exp\left(w_k^T x + w_{k_0}\right)}{\sum_{j=1}^{K} \exp\left(w_j^T x + w_{j_0}\right)},$$

where the denominator is the sum over all of the class discriminants outputs. This normalizes the output for each class relative to the likelihoods predicted by the other classes.

Mathematically, this implies that

$$\lim_{w_k^T x + w_{k_0} \to +\infty} P(C_k \mid x) = 1$$
$$\lim_{w_k^T x + w_{k_0} \to -\infty} P(C_k \mid x) = 0.$$

Learning Parameters. An appropriate loss function for logistic regression is cross-entropy loss. This is defined as

$$\mathrm{E}(w_k^T \mid x) = -\sum_t r^t \log y^t + (1 - r^t)\log(1 - y^t),$$

where $y^t = w_k^T x^t$ (for notational simplicity, it is assumed that the first column of X is a vector of ones, acting as the intercept). r^t is the actual class of the instance.

Therefore, the gradient of the loss function with respect to the model parameters is

$$\frac{dE}{dw_k} = -\sum_t (r^t - y^t)x^t.$$

This will be used in the gradient descent algorithm as the following update rule:

$$\Delta w_k = -\eta \frac{dE}{dw_k} = \eta \sum_t (r^t - y^t)x^t$$
$$w_k^{n+1} = w_k^n + \Delta w_k^n$$

This process is repeated iteratively until some convergence criteria is met, or a maximum number of iterations is exceeded.

Prediction. At prediction time, the outputs of each of the k discriminants are calculated, and the largest one is chosen. Because the outputs are normalized by the outputs of all of the other classes, the class that has the largest value is the most likely class for that instance.

2.2. **Adaline Network.**

Training. The Adaline Network[6] unit is quite similar to the logistic regression model. A linear discriminant is estimated via gradient descent. The difference is that rather than applying the logistic function to the output of the linear transformation, the Adaline Network tries to minimize the squared distance between its network outputs and the correct value. In other words, if the class instance is 0, the output of the linear transformation should be close to 0, and if the class instance is 1, the output of the linear transformation should be close to 1.

To train the network, the squared error loss function is used:

$$E(w, x^t) = \sum_t (w^T x^t - r^t)^2,$$

where w_k are the weights of the network (as above, assuming the first element of x^t is a 1 representing the intercept), and $r^t \in \{0, 1\}$ is the correct label. Next, the gradient of the loss function with respect to w^T must be calculated:

$$\frac{dE}{dw} = \sum_t (w^T x^t - r^t) x^t.$$

This leads to the following update rule:

$$\Delta W = \eta(r^t - w^T x^t)x^t$$
$$W^{n+1} = W^n + \Delta W^n$$

Prediction. At prediction time, the output of the network is discretized to produce a class, with values above .5 taken as a 1, and those below .5 as a zero. The raw output of the network can be used as a proxy for the prediction probability, which is leveraged in multiclass problems, where 1 vs. k classifiers are trained, and the class with the largest probablity is chosen.

3. Experimental Approach

The algorithms described above were applied to 5 real-world problems. As neither of the algorithms can handle null-values, those values were either dropped or transformed into one-hot encoded variables. Likewise, multi-valued discrete features are transformed to one-hot encoded variables.

Each of the datasets undergo feature transformation to scale everything to $[-1, 1]$, where all values are divided by the maximum absolute value in the column.

For each dataset, the algorithms are measured by their accuracy on the classification problem. They are also compared to a naive baseline model that predicts the class label mode over the training data.

Stratefied 5-Fold cross-validation is used to get a better estimate of the performance of each classifier on various out-of sample portions of the data.

4. Experiment Results

4.1. **Breast Cancer Dataset.** The Breast Cancer dataset[7] involves classifying tumors as malignant or benign based on characteristics of the tumors. This is a simple two-class problem, with continuous valued inputs, and so no data transformation is needed. The features are scaled using the max-scaler described above. The results of the experiments are shown in Table 1.

Algorithm	Average Accuracy
Naive Baseline	65.0%
Logistic Regression	95.6%
Adaline	96.1%

Table 1. Breast Cancer dataset experiment results.

4.2. **Glass Dataset.** The Glass Dataset[2] involves classifying the source of broken glass based on characteristics about the glass. This is a multi-class classification problem, and so a one-vs-all approach is used for the Adaline Network. Logistic regression handles that scenario naturally. The features are scaled using the max-scaler approach from above. The results can be found in Table 4.2.

Algorithm	Average Accuracy
Naive Baseline	35.5%
Logistic Regression	44.8%
Adaline	47.1%

Table 2. Glass dataset experiment results.

4.3. **Iris Dataset.** The Iris Dataset[1] presents a classification problem in which the species of iris flower must be classified based on attributes about the plant. All features are continuously valued, and so no encodings are needed there, but all features are transformed using the max-scaler. This is a multi-class problem, and so a one-vs-all approach is used for the Adaline Network. The results are shown in Table 4.3

Algorithm	Average Accuracy
Naive Baseline	33.3%
Logistic Regression	96.0%
Adaline	84.7%

Table 3. Iris dataset experiment results.

4.4. **Soybean Dataset.** The soybean dataset[4] has a classification problem in which the type of rot is classified based on features about the beans. The features are all discrete valued, and so all columns are one-hot encoded. The results are shown in Table 4.4.

Algorithm	Average Accuracy
Naive Baseline	36%
Logistic Regression	100%
Adaline	100%

Table 4. Soybean dataset experiment results.

4.5. **House Votes Dataset.** The house voting dataset[5] involves a classification problem where the party of the voter is predicted based on their votes. All of the features are one-hot encodings. Null values are encoded as their own boolean values. The results of the experiments are shown in Table 4.5.

Algorithm	Average Accuracy
Naive Baseline	61.4%
Logistic Regression	94.2%
Adaline	95.4%

Table 5. House Votes dataset experiment results.

5. Algorithm Behavior

In the experiments run, the algorithms perform comparably. They both outperform the baseline in all the experiments, indicating that the problems present classes that can be linearly seperated to varying degrees, but always exceeding a naive model that chooses the most popular class.

Training the algorithms was more cumbersome than expected, as tuning the learning rate and the number of iterations to run required some trial and error. However, it isn't too difficult to detect convergence, and both algorithms converge to similar performance levels.

6. Summary

In conclusion, this paper introduced two parametric techniques for learning linear discriminants. The paper introduced the training process via gradient descent. The algorithms were the dispatched on 5 real-world classification problems. Both algorithms outperformed the naive baseline on all problems. Both algorithms performed quite comparably in terms of accuracy on all of the problems.

References

[1] R.A Fisher. Iris. URL: `https://archive.ics.uci.edu/ml/datasets/Iris`.

[2] B. German. Glass identification data set. URL: `https://archive.ics.uci.edu/ml/datasets/Glass+Identification`.

[3] N.M. Laird G.M. Fitzmaurice. Multivariate analysis: Discrete variables (logistic regression). *International Encyclopedia of the Social and Behavioral Sciences*, pages 10221–10228, 2001.

[4] R.S.n Michalski. Soybean (small) data set. URL: `https://archive.ics.uci.edu/ml/datasets/Soybean+%28Small%29`.

[5] Jeff Schlimmer. Congressional voting records data set. URL: `https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records`.

[6] B. Widrow. An adaptive adaline neuron using chemical memistors. 1960.

[7] Dr. WIlliam H. Wolberg. Breast cancer wisconsin (original) data set. URL: `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29`.