Project #5: Linear Classifiers

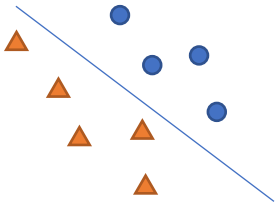
Paul Fornia | JHU 605.649
November 4, 2018

Abstract

I explore linear classification models, including Naïve Bayes, the Logistic Regression, and the Adaline artificial neural network. These methods appear very different on the surface, however, all three seek to divide data into classes along a hyperplane. The data sets considered look at breast cancer diagnoses, chemical characteristics of glass, physical characteristics of different species of the Iris, diseases among soybean samples, and vote records of several US house representatives. These all come from the University of California Irvine Machine Learning dataset repository. I test the hypotheses that a) all three models will outperform a simple baseline, b) the Logistic Regression and Adaline models will outperform Naïve Bayes, and c) the Logistic Regression and Adaline network will exhibit very similar levels of performance due to their mathematical similarity. In the course of running these experiments, I also find that the Logistic and Adaline models are more difficult to train, as they are sensitive to the learning rate and convergence criteria. Non-optimal hyperparameters can lead to suboptimal solutions, slow convergence, or even divergence.

Intro and Problem Statement

Linear classification models are simple, yet powerful techniques in the field of Machine Learning. They seek to find a hyperplane that divides the feature space along target class. In two dimensions, this can be easily visualized as a dividing line:



Although the purpose is to find a hyperplane that best divides the classes, the data need not be completely linearly separable, as is necessary with simpler approaches such as the perceptron. Naïve Bayes, Logistic Regression, and Adaline are all designed to handle noisy data in which some observations may fall on the wrong side of the optimal discriminant hyperplane.

I implement and test these three algorithms on five datasets. Each of these datasets lends to a two-class or multi-class classification problem. They have a mixture of binary, categorical, and numerical features. For simplicity and comparability, I convert all features to binary. First, I fill in any missing values with the most frequently occurring value for that feature. For the categorical variables, I perform one-hot-encoding (creating one binary variable for each possible value). For the numerical features, I first categorize them into equally populated quartiles, and then similarly perform one-hot-encoding on those quartile categorizations.

The Logistic Regression is designed for a two-class problem. I handle the multi-class datasets by running a "one-vs-all" logistic regression for each of the possible target classes. The prediction that chooses its corresponding class with the highest confidence will ultimately determine the final prediction. The Naïve Bayes can handle multi-class problems with no modification. The Adaline can be written with multiple output nodes known as a "multinet"; however, this is very similar in practice to the one-vs-all approach, since each output node gets its own set of weights. For simplicity, the Adaline code is structured similarly to the Logistic regression code, and effectively runs as a one-vs-all classification for each target class.

I test the following three hypotheses:

Hypothesis 1
For each of the five datasets, all three models will outperform the naïve baseline (always predicting the most frequent target class).

Hypothesis 2
For each of the five datasets, there will exist values for the *eta* (learning rate) and *epsilon* (convergence criteria) hyper-parameters for which the Logistic Regression and Adaline will outperform Naïve Bayes.

Hypothesis 3
For each of the five datasets, the Logistic Regression and Adaline models will exhibit a similar level of performance (assuming that each uses an approximately optimal set of hyperparameters).

The Algorithms

Naïve Bayes
Note: Description taken from Project 1 Summary

Naïve Bayes is based off of Bayes Theorem, and calculates prior and conditional probabilities by counting joint occurrences present in the training data set. Because of this, the training algorithm is actually just a process of counting up frequencies of observations.

The model is "trained" by counting

- Number of observations with each label in the target class (prior probabilities)
- For each feature:
    - For each unique factor in the feature:
        - Count number of observations with each label in the target class (joint occurrences, can be zero)

Since we want to know the most probable class, we are seeking the class label *c* that maximizes the probability of *c* given all the data *D* that we know about an observation. We transform this probability with Bayes Theorem:

$$P(c|D) = \frac{P(D|c)P(c)}{P(D)}$$

P(D) is independent with respect to the argmax variable, so we can drop it. In simpler terms, the data is what the data is, so we do not need to adjust all our probabilities to account for whether the data was likely to occur.

Next, *P(D|c)* is expanded to the joint probability of all feature values making up the data, and is written $P(d_1, d_2, \dots, d_m \mid c)$.

Finally, the algorithm assumes conditional independence, and so the joint probability can be re-written as a product:

$$argmax_c P(c|D) = argmax_c P(c) \prod_{i=1}^{m} P(d_i|c)$$

Now, each of these terms is easily derived from the counts obtained in the training process. One is added to all counts to "smooth" the training output. This prevents any terms from completely disappearing when a feature/label combination is not seen.

Logistic Regression

Logistic Regression is very similar to Ordinary Least Squares (OLS) regression. It provides a vector of weights that is multiplied by each feature of an observation to obtain a prediction value for that observation's target value. But unlike OLS, it has one additional transformation step, which uses the logistic function to transform this linear combination into a probability value from 0 to 1.

Gradient descent is used to train the optimal weight vector.

$$w_{t+1} = w_t + \eta \sum_k x^k (y^k - logit(w_t^T x))$$

Where the logistic function is defined as:

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

Adaline

Like Logistic Regression, the Adaline consists of a vector of weights, which are used to calculate a linear combination of the features. The activation function of Adaline is a bit simpler than the sigmoid function of Logistic Regression: if the linear combination is negative, the prediction is the negative class, while a positive linear combination indicates a positive class prediction.

The training of the Adaline is guided by the Widrow-Hoss rule. While similar in spirit to the Logistic iterative update, it has some small differences, and is not based on a clean derivative (since the "sign" activation function is not differentiable). It is also independent of the activation function, which is used only after training is complete.

$$w_{t+1} = w_t + \eta(y - w_t^T x) * x$$

Experimental Design

For all of these models (and all five datasets), I measure performance based on a simple accuracy measure – number of correct predictions divided by total number of observations. To protect against unbalanced datasets, I compare these metrics to the naïve method of always predicting the most common class.

As always, metrics are calculated on "out-of-sample" test sets, utilizing a 5-fold cross validation approach. For each fold, I use the training data (four-fifths of the data) to train a model. I then use that model to classify the observations of the remaining test data (one-fifth). After all five folds are complete, I calculate an average performance metric across the folds.

Results

Dataset	"Naïve" Baseline	Naïve Bayes	Logistic Regression* (eta/epsilon)	Adaline Neural Net* (eta/epsilon)
Breast Cancer	65.5%	**65.5%**	**96.6%** (0.01/0.01)	**95.6%** (0.01/0.01)
Glass	35.5%	**43.5%**	**60.3%** (0.05/0.001)	**60.3%** (0.05/0.001)
Iris	33.3%	**95.3%**	**96.0%** (0.01/0.01)	**96.0%** (0.01/0.001)
Soy	36.2%	**69.8%**	**100%** (0.1/0.001)	**100%** (0.1/0.001)
Vote	61.4%	**90.1%**	**95.2%** (0.01/0.01)	**92.0%** (0.01/0.01)

*Logistic Regression and Adaline both required some manual tuning and experimentation of the learning rate and convergence criteria.

- Hypothesis #1 is confirmed – All models outperformed the naïve baseline for all five datasets (although Naïve Bayes is very nearly a tie for the Cancer dataset).
- Hypothesis #2 is confirmed – Logistic Regression and Adaline outperform Naïve Bayes for all datasets.
- Hypothesis #3 is confirmed – Given approximately optimal hyperparameters, Adaline and Logistic Regression perform approximately equally on all five datasets.

Algorithm Behavior and Conclusions

The Adaline and Logistic methods were a bit tricky to tune. Take the Cancer dataset for example. The table above shows convergence based on an epsilon of 0.01 (the threshold of change from one weight vector to the next at which the algorithm terminates). With an epsilon of 0.001, the algorithm took an order of magnitude longer to run, and the results were nearly indistinguishable. In contrast, increasing the epsilon to 0.05 made the performance go down so significantly that it could barely be distinguished from the naïve baseline rate.

Similar can be said of tuning the learning rate. The soy data required an unusually high learning rate, and a very small termination criterion. With a slightly higher epsilon, the prediction accuracy was worse than the baseline, and with a slightly higher eta of just 0.2, the gradient descent diverged and only terminated after a fixed maximum number of iterations. However, with some trial-and-error, the selected eta and epsilon ran relatively quickly, and produced a model with 100% accuracy on all five test sets.

In general, Epsilon seemed to tradeoff performance with accuracy. Too high, and the model may not have iterated a sufficient number of times. Too low, and the model may take unnecessarily long to converge.

Generally, the learning rate eta seemed to mostly just guide speed of convergence, and hence performance. Too low, and the models would not be allowed to change fast enough. When eta is *slightly* too high, then the weights may change too quickly, "overshooting" their optimal, increasing the number of iterations needed. When eta is *far* too high, the weights diverge away from their optimal state, and the resulting model is useless.

Summary

Three different linear classification algorithms are explored. Naïve Bayes outperforms a simple baseline, predicting the most common class, for all five datasets. Logistic Regression and Adaline both outperform Naïve Bayes, and perform similarly to each other, consistent with their similar structure.

References

Littlestone, N. *Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm.* Machine Learning (1988) 2: 285-318. https://doi.org/10.1007/BF00116827

Alpaydin, Ethem. Introduction to Machine Learning (Adaptive Computation and Machine Learning series). (page 248) The MIT Press. Kindle Edition.

Widrow, B., Hoff, M. E. (1960). *Adaptive Switching Circuits.* Stanford Electronics Laboratories, Stanford, CA.

University of California Irvine *Machine Learning Repository* https://archive.ics.uci.edu/ml/index.php Accessed 2018-11-04.