Logistic Regression for White Wine Quality and Blood in Kidney Classification (MP1)

Norman Kong
Department of Electrical, Computer and Software Engineering
McGill University
Montreal, Quebec
norman.kong@mail.mcgill.ca

Yinuo Li
Department of Materials Engineering
McGill University
Montreal, Quebec
yinuo.li@mail.mcgill.ca

Yicheng Yuan
Department of Electrical, Computer and Software Engineering
McGill University
Montreal, Quebec
yicheng.yuan@mail.mcgill.ca

Abstract

This project is aimed to implement linear classifier logistic regression in Python to classify two groups of data. Each group of data is divided into train data and test data while train data is further divided into train and validation sets based on the K-fold method. The performance of the model is evaluated based on the accuracy of the test data in the end. In the blood-in-kidney example, dropping features by minimum weight was applied, giving a final accuracy of 0.748. For the white wine quality dataset, feature insertions were selected with a final accuracy of 0.70. The change of the model precision due to the modification of the learning rate is also studied afterward.

1 Introduction

Machine learning (ML) has been widely used in a lot of applications such as health care and food industry which has been introduced as examples in this project. It is a field of study modeling computational algorithms to transform the known input information to desired unknown information. Supervised learning can be divided into two general categories: regression and classification. Regression predicts continuous value while classification predicts discrete classes.

In this project, linear classifier logistic regression as a classification algorithm is implemented to classify two datasets, which are the white wine quality and blood in kidney. The datasets will be divided into train data and test data. While train data is used to obtain the model, test data will be used to evaluate the performance of the model. The performance of train data is quantified by the 10-fold cross validation. Furthermore, the learning rate will be modified and evaluated to find the optimum learning rate to get the lowest iterations and highest accuracy for the model.

1st Assignment for ECSE 551: Machine Learning for Engineers (W2023).

The quality of wine will be predicted from its compositions in the while wine quality dataset, which can be used to improve the quality of wine in the food industry. The blood in kidney dataset measures two health features to evaluate whether there will be blood in the kidney, giving the information on inflation.

The model are implemented from scratch with basic libraries such as numpy and panda. However, the existing models can be found in Scikit-learn library. A comparison of performance will be made between the implemented model and the model directly from the Scikit-learn library.

2 Datasets

The datasets for this experiment were provided by Prof. Armanfard. They were given in normalized form and so this preprocessing step was not necessary in our experiments.

2.1 White Wine Quality Datasets

This dataset contains 1599 samples. We would like to classify samples into two classes: low-quality (class 0) and good-quality (class 1). There are 744 samples for class 0 and 855 samples for class 1. This is close to a uniform distribution and so this dataset exhibits high entropy. The order of the 10 attributes for each sample are: Alcohol, Malic acid, Ash, Alkalinity of ash, Magnesium, Total phenols, Flavanoids, Nonflavanoid phenols, Proanthocyanins and Hue. These are believed to affect wine quality and hence are the features on which we will train the model. The last column denotes the class for the sample.

2.2 Blood in Kidney Datasets

This dataset consists of 330 samples. We would like to classify samples into two classes: blood is in the kidney (class 1) or not (class 0). The data is split evenly among classes (160 samples per class), meaning this dataset exhibits maximum entropy. The order of the 9 attributes for each sample are: Pregnancies (number of times pregnant), glucose (plasma glucose concentration a 2 hours in an oral glucose tolerance test), BloodPressure (diastolic blood pressure in mm Hg), Heart Rate, SkinThickness (triceps skin fold thickness in mm), Insulin (2-Hour serum insulin in mu U/ml), BMI (body mass index), DiabetesPedigreeFunction (diabetes pedigree function) and Age (in years). The last column denotes the class for the sample.

3 Methods

3.1 Data Exploration

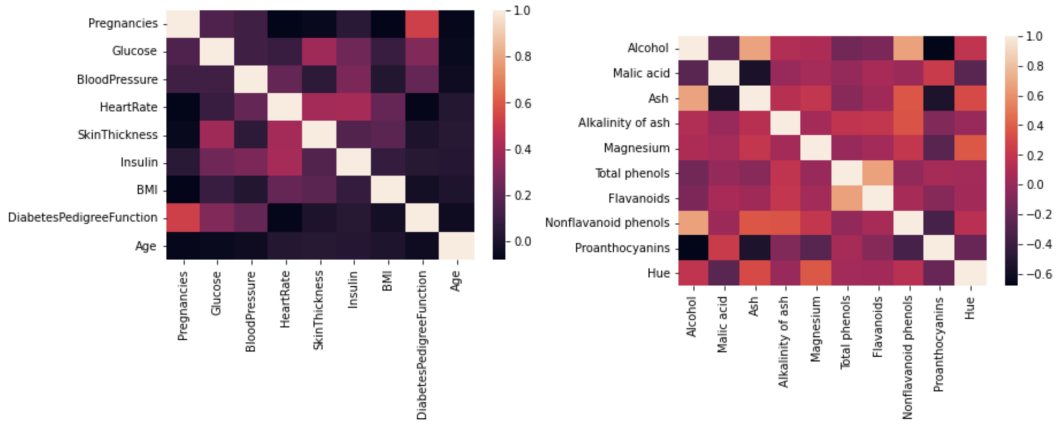
Common practice in machine learning experiments is to do perform basic preliminary data analyses. To avoid an overly long report, we have omitted most of our exploratory data visualizations (please refer to the Google Colab for more). However, we have included the heatmaps for the kidney and wine datasets (respectively) below as this was important in inspiring us and providing motivation for many of our improvement attempts.

We see that certain features such as alcohol and ash are high correlated; this leads us to the question of whether or not one of these features could be redundant, since it may behave similarly to the feature it is highly correlated with.

3.2 Logistic Regression

The logistic regression model is constructed with the following fields: learning rate, epsilon, max iterations, verbose, and draw_loss. The epsilon and max iterations are both used as terminating conditions for the gradient descent of logistic regression. The draw_loss term determines whether the loss function should be drawn.

The cost_fn method in the logistic regression model calculates the cost function. The fit method is implemented following the formula in lecture 6 of ECSE 551. However, a tiny change is we calculate the delta term with the division over N. That's because the cost is the mean over n terms



[1]. However, since the learning rate is multiplied before the delta, so we found the division over N is not necessary, instead, we can adjust the learning rate. One effect of this choice is that the learning rate is increased N times to compensate for the division of the delta by N, and show the pattern of iterations. So when drawing the plot of iterations vs the learning rate, the learning rate has to be divided back.

When we train and predict using the logistic regression model we created, we can first call the fit method with train input and output, as well as the learning rate which we choose a default number of 0.1 after attempts over multiple numbers. The fit method can update the weight vector of the model, and then we can use the predict method to predict the output using the test data. We need a method to split the test and train data in this case.

The train_test_split method is implemented. Given a Panda data frame, we can split 70 percent of the data as a training set and 30 percent as a testing set. The Accu_eval method counts the ratio of correctly predicted data.

3.3 Hypotheses For Improvements

Training and evaluating the logistic regression model with standard hyperparameters:

```
max_iters = 1e5, lr=0.1
```

on all features is our baseline. Methods to possibly improve the model's performance is by training it on a subset of features and by adding features.

3.3.1 Training on a Subset of Features

To choose our subset of features, we attempted the following approaches:

Method 1: Drop Feature For Smallest Weight We first tried dropping the feature with the minimum absolute weight. The motivation for this is that the small magnitude of the weight suggests it is "unimportant" in predicting the class.

Method 2: Drop Half the Feature For Smallest Weights What if we push the logic of method 1 even further? We attempt to fit the model $\frac{n}{2}$ times; after each fit, we drop the lowest weight (where n corresponds to the number of features).

Method 3: Dropping Highly Correlated Features We try dropping a feature (say, X) that is highly correlated to another feature; the motivation for this is that the high correlation may be an indication that X behaves similarly to another feature and so X may be redundant.

Method 4: RFE We will see that method 2 in fact leads to similar performance as our baseline for both datasets. This is perhaps unsurprising since our guess as to which feature is unimportant is a

	Methods						
	Baseline	1	2	3	4	5	final
kd	71.30	72.17	72.17	68.79	70.00	70.00	74.78
ww	72.17	72.79	71.80	55.41	72.16	73.06	70.00

Table 1: Accuracy table displaying accuracy scores across baseline, attempted improvements and the final accuracy score. Values represent accuracy percentages (correct classifications over all classifications).

	Methods					
	Baseline	1	2	3	4	5
kd	29.3	29.5	41	36.5	24.6	71
ww	255	187	141	151	97	331

Table 2: Table of runtimes over various attempted improvement methods. Displays runtimes for each method and the baseline. Values represent runtime in seconds.

naive guess. What if we allow a proper algorithm to decide what features to exclude? We explored how to use RFE as an approach to feature selection. RFE recursively removes features from the dataset and builds a model using the remaining features. We implement RFE by recursively removing one feature from the dataset and calculating the mean square error. After we find out which feature dropped gives the minimum mean square error, we will know that feature is the least important one. The least important features will then be eliminated, and the process is repeated until the desired number of features is reached.

3.3.2 Training with an Added Feature

Method 5: Basis Expansion of Features There are several ways to modify our existing features to create new ones. As discussed in lecture, there's a transformation of variables, basis expansions, interaction terms, etc. We will implement basis expansion on the features corresponding to the 3 highest weights by squaring them.

4 Results

4.1 Accuracy

Shown in Table 1 is the accuracy table, which displays accuracy scores across baseline, attempted improvements and the final accuracy score. Values represent accuracy percentages (correct classifications over all classifications).

4.2 Runtime

As expected, training and testing of the (much larger) wine dataset is computational more expensive. The runtimes are shown in Table 2, which displays runtimes for each method and the baseline. Values represent runtime in seconds. Note that part of the runtime shown below include printing information such as runtime, weights, etc., which typically slow the code down. However, since this was consistent across all runtime measurements, it is still fair to compare runtimes for various methods.

4.3 Learning Rate

Shown in Figure 1 are the graphs of the learning rate versus accuracy (top) and learning rate versus number of iterations (bottom), for the kidney (left) and ww (right) datasets, respectively.

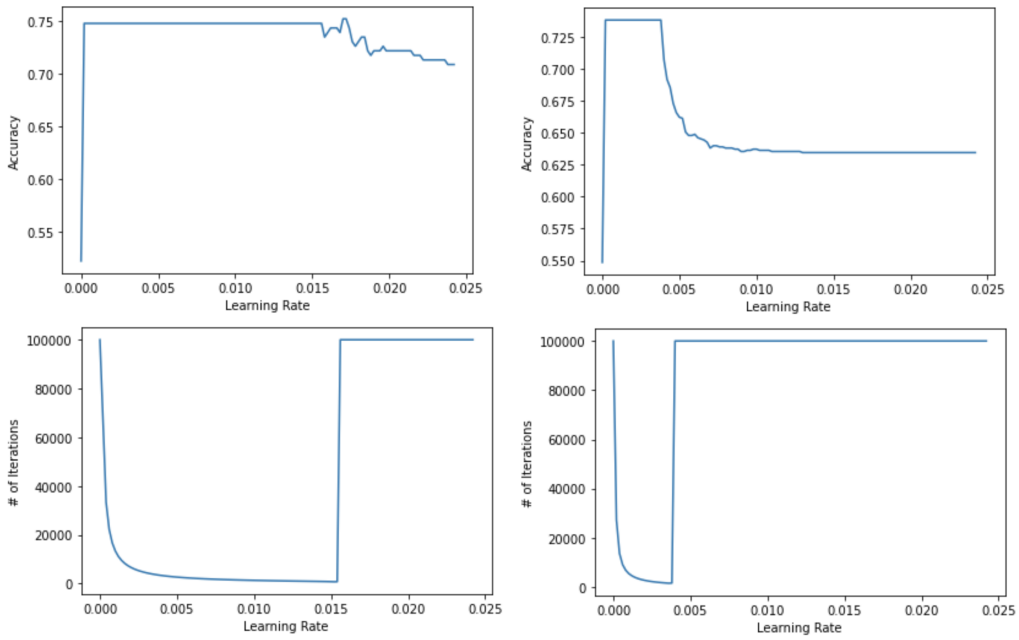


Figure 1: The graphs of the learning rate versus accuracy (top) and learning rate versus number of iterations (bottom), for the kidney (left) and ww (right) datasets, respectively.

5 Discussion and Conclusion

5.1 Accuracy

There are two types of feature selection methods, which are unsupervised feature selection and supervised feature selection respectively. For the unsupervised feature selection, correlation matrix is found to determine whether there are features with high correlations. Then the feature could be deleted in order to avoid overfitting and to increase the computation efficiency. Without feature selection, the accuracy on the training data is 0.729 for white wine quality and 0.713 for blood in kidney. The threshold of the correlation is set to be 0.5 for blood in kidney dataset and 0.65 for white wine quality dataset. It is relatively low compared to usual cases when unsupervised feature selection is applied. But in this case, as in both datasets the features are not very much correlated, a relatively low threshold is applied just to determine how this method will influence the accurracy. One feature is dropped for the blood in kidney dataset and four were dropped for the white wine quality dataset. After dropping those features the blood in kidney example gives an accuracy of 0.688 and the white wine quality example gives an accuracy of 0.554. Both of those accuracy decreased, claiming that this feature selection method is not applicable in this case, due to the low correlation among features.

5.2 Runtimes

As expected, the runtimes for the kd dataset are much shorter (expected because it is much smaller, so training is far less computationally expensive). Furthermore, methods 1-4 reduce the amount of features in our data, which helps explain the decrease in runtime for the wine dataset. However, we note that while the wine dataset decreases for most methods, the kidney dataset in fact increases for several methods. A possible explanation for this is that although we are reducing the number of features, the saved time from removing features is less apparent on a smaller dataset since its computation was already small; and in fact, the preprocessing required to select and remove features becomes more significant.

5.3 Learning Rate

Learning rate needs to be determined to define how quickly the model could calculate the optimal weights. Too large of a learning rate will lead to difficulties for functions to converge, while too small a learning rate will result in too many steps. As shown in the figures below, the number of iterations will first decrease then increase as learning rate increases. The accuracy will first increase, reaches a plateau then decrease as learning rate decreases. The optimal learning rate should be taken as when the lowest number of iterations and maximum accuracy appear. In the blood in kidney dataset, the optimum learning rate is 0.015 while the accuracy reaches around 0.75. In the white wine quality dataset, the optimum learning rate is 0.0038 with an accuracy of 0.73.

6 Statement of Contributions

Coding and writing tasks were shared equally among all members.

7 References

[1] "Gradient descent in linear regression," GeeksforGeeks, 11-Jan-2023. [Online]. Available: https://www.geeksforgeeks.org/gradient-descent-in-linear-regression/. [Accessed: 19-Feb-2023].

8 Appendix

We have attached a PDF of our Google Colab for your convenience in reading the code. Please see the PDF for extra data exploration, loss graphs and more. Equivalently, these are also available in the Google Colab itself.