EE3-23: Coursework for Introduction to Machine Learning

Douglas Brion
Imperial College London
CID: 01052925
db1415@ic.ac.uk

1. Introduction

This report examines different approaches to predicting wine quality using the Wine Quality [5] dataset. This is a large dataset containing 4898 samples of white wines and 1599 of red. Throughout this report wine quality is examined using regression, the standard approach when modelling continuous data, attempting to predict the quality of a wine from various input parameters. Multiple learning methods are implemented, discussed and compared in order to obtain a predictor with the smallest test error possible.

All code for this report was written in Python using the libraries Pandas [4] for importing and manipulating data and Tensorflow [1] for creating and training models to predict the data.

2. Data Preparation

Ad red and white wine have different tastes, it was decided to learn on both separately as they have a different chemical composition. The white wine dataset contains 4898 samples and the red 1599. The objective in this report is to establish how well the quality of each wine can be predicted using the datasets.

It should be noted that this data is likely to be unreliable with a large amount of noise as quality is measured with a human sense, taste a very personal sense which can range widely person to person.

The data provided was not normalised, therefore each attribute was normalised with respect to it's mean and standard deviation. Outliers over a threshold of for each attribute were removed to reduce noise in the dataset for training. Table 1 and Table 2 show the normalised physiochemical attributes for the datasets with outliers removed.

The spread in quality for both data sets is not ideal with the histograms Fig 1 and Fig 2 illustrating the distribution. The quality of the wine is graded on a scale from 0 (really bad) to 10 (extremely good) with the datasets containing six/seven classes (3 to 8/9).

The goal of this project will be to find a predictor with the best regression performance.

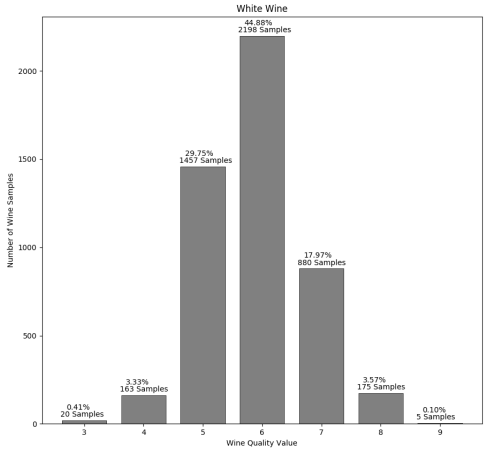

Figure 1. The histogram for white wine qualities.

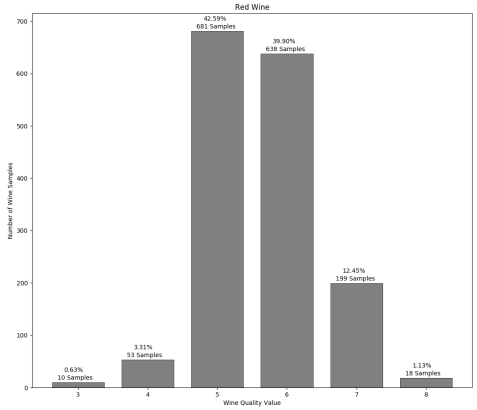

Figure 2. The histogram for red wine qualities.

Attributes	Mean	Std	Min	Max
fixed acidity	-0.003	0.989	-3.62	4.557
volatile acidity	-0.012	0.961	-1.967	4.979
citric acid	-0.010	0.963	-2.762	4.758
residual sugar	-0.002	0.986	-1.142	4.971
chlorides	-0.078	0.662	1.683	4.954
free sulfur dioxide	-0.010	0.956	-1.959	4.892
total sulfur dioxide	-0.003	0.992	-3.044	4.839
density	-0.005	0.971	-2.313	2.984
pH	0.000	1.000	3.101	4.184
sulphates	-0.001	0.997	-2.365	4.996
alcohol	0.000	1.000	-2.043	2.995

Table 1. The white wine attribute statistics after removal of outliers and normalisation.

Attributes	Mean	Std	Min	Max
fixed acidity	0.000	1.000	-2.137	4.355
volatile acidity	-0.004	0.989	-2.278	4.481
citric acid	0.000	1.000	-1.391	3.744
residual sugar	-0.053	0.764	-1.163	4.584
chlorides	-0.096	0.552	-1.604	3.880
free sulfur dioxide	-0.003	0.991	-1.423	4.985
total sulfur dioxide	-0.009	0.967	-1.231	3.604
density	0.000	1.000	-3.539	3.680
pH	0.000	1.000	-3.700	4.528
sulphates	-0.033	0.879	-1.936	4.142
alcohol	0.000	1.000	-1.899	4.202

Table 2. The red wine attribute statistics after removal of outliers and normalisation.

2.1. Learning approach

For consistency all the different regressions performances of each model with varying parameters and training cost functions will be measured using the same error metric, the Huber Loss function. This is a good loss function for this regression problem as it is robust to outliers, for if the difference between the real and predicted value is small it will be squared, if large, the absolute value will be taken.

$$L_H = \begin{cases} \sum_{i=1}^{n} \frac{1}{2}(y_i - f(x_i))^2, & for|y_i - f(x_i)| \leq \delta. \\ \sum_{i=1}^{n} \delta|y_i - f(x_i)| - \frac{1}{2}\delta^2, & otherwise. \end{cases} \tag{1}$$

A robust procedure for estimation, *k*-fold cross validation [3], where data is divided into *k* parts and with one subset tested at a time and remaining data used for training. This methods results in all data being used for both training and testing, however requires a longer computation time.

3. Baseline Predictors

Several baseline predictors have been implemented and trained on the data.

A linear regression program was written in Python using Tensorflow [1]. This linear model used an iterative method to alter the weights, with multiple loss functions tested. This proved useful as a framework for the more advanced algorithms implemented later on.

For linear regression 2 basic loss functions were implemented first.

3.1. L1 loss function

The L_1 loss function, least absolute deviations, tries to minimise the absolute difference between the predicted and real values. The sum of all the differences for all the samples can be described as follows:

$$L_1 = \sum_{i=1}^{n} |y_i - f(x_i)| \tag{2}$$

This is a fairly robust loss function which is not that affected by outliers.

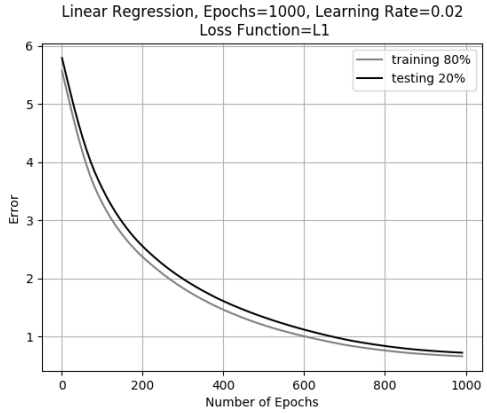

Figure 3. Loss Function = $\sum_{i=1}^{n}|y_i - f(x_i)|$, Training Error = 0.321, Testing Error = 0.373

3.2. L2 loss function

The L_2 loss function, least square error, minimises the square difference between the predicted and real values. This value is much large than that of L_1 loss and therefore is more affected by outliers, however, will optimise the predictor faster.

$$L_2 = \sum_{i=1}^{n} (y_i - f(x_i))^2 \tag{3}$$

Notice how the predictor using the L_2 loss function in 4. minimises the error much faster than the L_1 loss in 3 and as

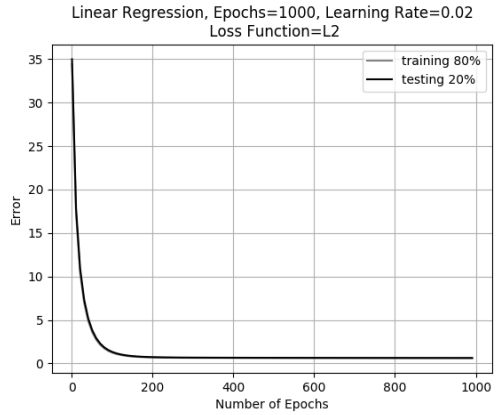

Figure 4. Loss Function = $\sum_{i=1}^{n}(y_i - f(x_i))^2$, Training Error = 0.271, Testing Error = 0.276

outliers have been removed in data preparation the L_2 loss is quite robust.

As the epochs are increased the optimiser is able to reduce the loss until no improvements can be made. The learning rate also greatly affects the rate at which the loss is minimised. However, if the learning rate is too high the predictor will not improve on the training data, as can be seen in 5.

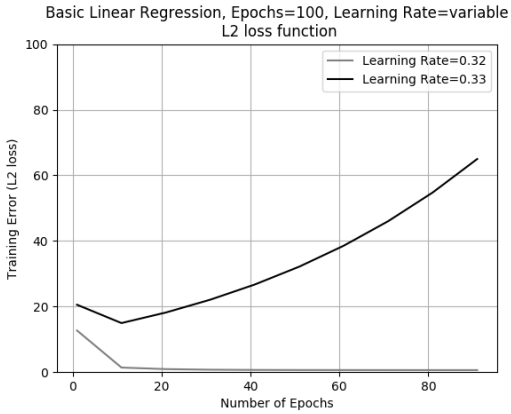

Figure 5. Training Error = $\sum_{i=1}^{n}(y_i - f(x_i))^2$ (L_2 loss), Epochs = 100, Learning Rate = variable

3.3. Improvements

These relatively simple models can be improved upon by adding regularisation. Regularisation is a technique used to prevent over fitting the data. Both L_1 and L_2 regularisation have been implemented, being the sum of the weights and sum of the square weights respectively. The regularisation term is added to the loss function, the terms for L_1 and L_2 look as follows:

$$L_{1reg} = \lambda \sum_{i=1}^{n} |w_i| \quad (4) \qquad L_{2reg} = \lambda \sum_{i=1}^{n} w_i^2 \quad (5)$$

4. More Advanced Algorithms

More advanced algorithms such as NNs and SVMs achieve high performance due to their non-linear learning capabilities. However due to this, these complex models are more likely to over-fit, losing the ability to generalise when given new data to test. Both SVMs, NNs and Elastic Regularisation have multiple hyper-parameters which need to be adjusted in order to find the best predictor.

4.1. Neural Network

The neural network implemented as a predictor for this project had 3 layers, 1 input, 1 hidden and 1 output. A single hidden layer was chosen as an increase in hidden layers did not result in a significant increase in predictor performance although it did lengthen training time.

The number of nodes in the hidden was chosen using a heuristic of the number being the mean of the input and output layers, in this case resulting in 6 nodes.

Each layer could be assigned when the network is constructed to have a specific activation function such as: TanH, Sigmoid, ReLU, SeLU and Softmax. It was found that during training a hidden ReLU resulted in the best out of sample test error and therefore was chosen.

Neural networks have an advantage over standard linear regression that they can model non-linearities automatically however they are more likely to over-fit the data so observing the out of sample error is especially important.

The neural net used a gradient descent optimiser to minimise the loss function, and either L_1, L_2 or no regularisation can be selected for training.

4.2. Support Vector Regression

The goal for a support vector regression (SVR) [2] predictor is to find a predication $f(x)$ like our other predictors however it should have at most ϵ deviation from the real value y_i for all the in sample data.

The loss function for SVR is as follows:

$$L_{SVR} = max(0, \sum_{i=1}^{n} |y_i - f(x_i)| - \epsilon) \quad (6)$$

This loss function is known as the hinge loss.

4.3. Elastic Net Regularisation

The elastic net regression is a regularised method combining the L_1 and L_2 penalties of the lasso and ridge methods. A hyper-parameter, α between 1 and 0 controls how much of L_1 and L_2 penalisation is used.

5. Results

In order to evaluate the possible models to find the best predictor, 10-fold cross validation was run for each model. Results graphs will illustrate the error of the model over the epochs with the final error for that models loss function shown as 'final error' in the legend. The final model after learning is then tested using Huber Loss 1 also shown in the legend. This is used to compare the various models against each other after training.

5.1. Linear Regression

For linear regression it was found that regularisation did not improve the performance of the predictor. As regularisation is used to help prevent over-fitting, and as basic linear regression is not as prone to over-fitting the data it was found that adding regularisation did not improve the performance.

5.2. Neural Networks

It was found that ReLU for both hidden and output layers was by far the most effective activation function. This may be because the ReLU function does not suffer from the vanishing gradient problem unlike functions such as tanh and sigmoid where the updating of weights can be prevented by a tiny gradient.

It was also found that regularisation greatly improved the performance of neural net predictors. After running extensive tests with varying parameters and comparing the huber error of each of them, the best performing networks for both the white and red wine data sets can be seen below. Notice how Red wine has a smaller error as the quality is concentrated at values 5 and 6, this also may explain why L_2 regularisation fairs better.

Regularisation is especially important for the neural nets as they have a tendency to over-fit. However the regularisation parameter, λ has to be tuned correctly to avoid not fitting the data at all.

Neural nets have a disadvantage over other methods proposed in this report as they may arrive a local minima during the learning phase, a problem algorithms such as SVRs do not encounter.

5.3. Support Vector Regression

5.4. Elastic Net Regression

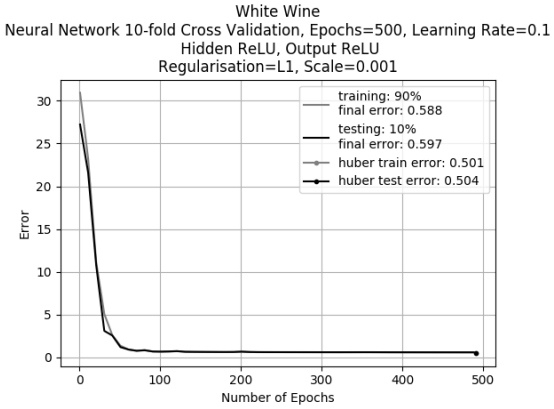

Figure 6. Best Neural Network White Wine predictor.

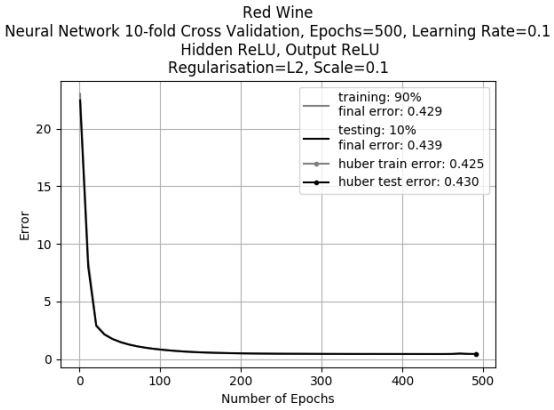

Figure 7. Best Neural Network Red Wine predictor

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[3] S. M. Cross-validatory choice and assessment of statistical predictions., 1974.

[4] W. McKinney. pandas: a foundational python library for data analysis and statistics.

[5] F. A. T. M. P. Cortez, A. Cerdeira and J. Reis. Modelling wine preferences by data mining from physicochemical properties., 2019. Face and Gesture submission ID 324. Supplied as additional material `fg324.pdf`.