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 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.

The objective of this report is to identify the predictor with the best regression performance.

2. Data Preparation

As red and white wine have different tastes, it was decided to learn on them separately as they differ in chemical composition. 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 and noisy for quality was measured with a human sense, taste, a very personal metric which can range widely from person to person.

The data was normalised with respect to it's mean and to a standard deviation of one. Outliers over a threshold of for each attribute were removed to reduce noise in the dataset for training. Tab 1 shows the normalised physiochemical attributes for the datasets with outliers removed.

The spread in quality for both data sets is not ideal for learning 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 red wine dataset is especially concentrated in the quality values 5 and 6.

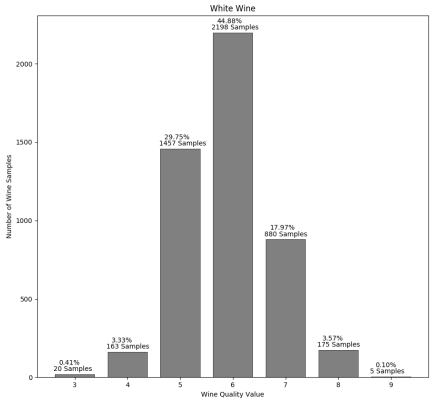

Figure 1. The histogram for white wine qualities.

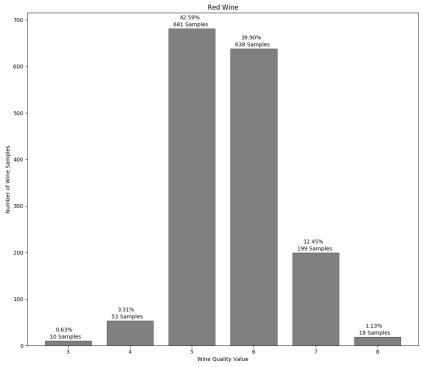

Figure 2. The histogram for red wine qualities.

2.1. Learning approach

For consistency the performance 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

Attributes	Mean		Std		Min		Max	
	White	Red	White	Red	White	Red	White	Red
fixed acidity	-0.003	0.000	0.989	1.000	-3.62	-2.137	4.557	4.355
volatile acidity	-0.012	-0.004	0.961	0.989	-1.967	-2.278	4.979	4.481
citric acid	-0.010	0.000	0.963	1.000	-2.762	-1.391	4.758	3.744
residual sugar	-0.002	-0.053	0.986	0.764	-1.142	-1.163	4.971	4.584
chlorides	-0.078	-0.096	0.662	0.552	1.683	-1.604	4.954	3.880
free sulfur dioxide	-0.010	-0.003	0.956	0.991	-1.959	-1.423	4.892	4.985
total sulfur dioxide	-0.003	-0.009	0.992	0.967	-3.044	-1.231	4.839	3.604
density	-0.005	0.000	0.971	1.000	-2.313	-3.539	2.984	3.680
pH	0.000	0.000	1.000	1.000	3.101	-3.700	4.184	4.528
sulphates	-0.001	-0.033	0.997	0.879	-2.365	-1.936	4.996	4.142
alcohol	0.000	-0.000	1.000	1.000	-2.043	-1.899	2.995	4.202

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

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.

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

A robust procedure for estimation was used, *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. LASSO loss function

The LASSO loss function 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:

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

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

3.2. Ridge loss function

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

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

Notice how the predictor using the Ridge loss function in 8 minimises the error faster than the LASSO loss in 7. Additionally, as outliers have been removed in data preparation the Ridge loss is also quite robust.

As the epochs are increased the optimiser is able to reduce the loss until no substantail improvements are 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 might not improve in training and may get worse, as can be seen in 3. (see **Appendix** to compare LASSO and Ridge)

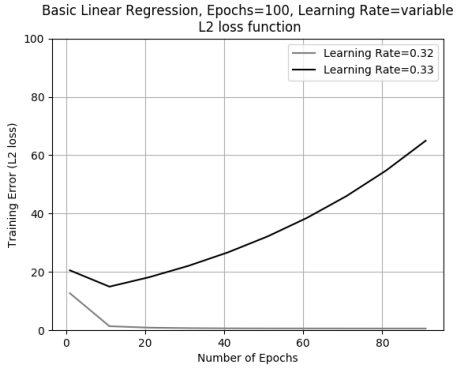

Figure 3. Training Error = $\sum_{i=1}^{n}(y_i - f(x_i))^2$ (L_2 loss), Epochs = 100, Learning Rate = variable

3.3. Regularisation

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, L_2 and Elastic Net regularisation have been implemented, being the sum of the weights and sum of the square weights and combination of the two respectively. The regularisation term is added to the loss function, the terms for L_1 and L_2 look as follows:

$$L_1 = \lambda \sum_{i=1}^{n} |w_i| \quad (4) \qquad L_2 = \lambda \sum_{i=1}^{n} |w_i|^2 \quad (5)$$

The elastic net regularisation combines the L_1 and L_2 penalties of the methods described in Eq 4 and Eq 5. A hyper-parameter, α between 1 and 0 controls how much of L_1 and L_2 penalisation is used.

$$L_{elastic} = \sum_{i=1}^{n} \lambda(\alpha|w_i| + (1-\alpha)|w_i|^2) \quad (6)$$

The computation time for elastic net regularisation is longer than standard LASSO or Ridge due to the additional hyper parameter α.

4. More Advanced Algorithms

More advanced algorithms such as Neural Networks can 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. Linear Support Vector Regression and Neural Networks have multiple hyper-parameters which need to be adjusted in order to find the best predictor.

4.1. Neural Network

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 network implemented as a predictor for this report consisted of 3 layers: input, hidden and output. A single hidden layer was chosen as an increase in hidden layers did not result in a superior predictor performance, although computation time was lengthened.

The number of nodes in the hidden was heuristically chosen as the mean of input and output layers, in this case resulting in 6 hidden nodes.

Whilst being constructed, each layer in the network can be set to a specific activation function such as: TanH, Sigmoid, ReLU, SeLU and Softmax. It was discovered during training that hidden and output ReLU layers resulted in the best out of sample test error this activation function was chosen.

The neural net used a gradient descent optimiser to minimise the loss function, and either L_1, L_2 or no regularisation was 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:

$$L_{SVR} = max(0, \sum_{i=1}^{n} |y_i - f(x_i)| - \epsilon) \quad (7)$$

This loss function is known as the hinge loss.

5. Results

In order to evaluate the possible models to find the best predictor, 10-fold cross validation was run for each model (90% training, 10% testing). 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. The plots are the average cross validation errors of the predictor after each 10 epochs.

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.

Linear regression with both LASSO (L_1) and Ridge (L_2) loss functions is successful both with and without regularisation for both white and red wine.

For L_1 the huber loss test error consistently was approximately 0.2 in the red wine dataset, with and without regularisation after 500 epochs with a learning rate of 0.05. The results for the L_2 loss function were very similar, however, the error was decreased significantly faster due to the square term.

A small regularisation term proved to reduce the error for the ridge loss function.

Elastic regularisation with an L_2 loss function and parameters $\alpha = 0.5, \lambda = 0.01$ produced the best performance overall with a huber loss after training of 0.171 for red wine. It was found similar to lasso and ridge regularisation large parameter values worsened the final error causing underfitting.

SVR when run with the elastic net proved to perform worse than without it. The combination of ϵ and the under-fitting from the regularistion resulted in a worse test error.

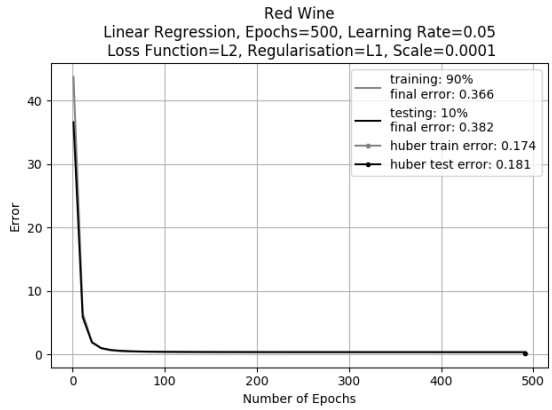

Figure 4. Red Wine - A good linear regression predictor with Ridge loss function and regularisation.

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

The linear support vector regression approach had quite similar results to basic linear regression with LASSO or Ridge loss functions, although the results appeared to be more consistent. It was also found that having a large regularisation for SVR worsened the predictors performance as the data was under-fitting. The huber test loss after training

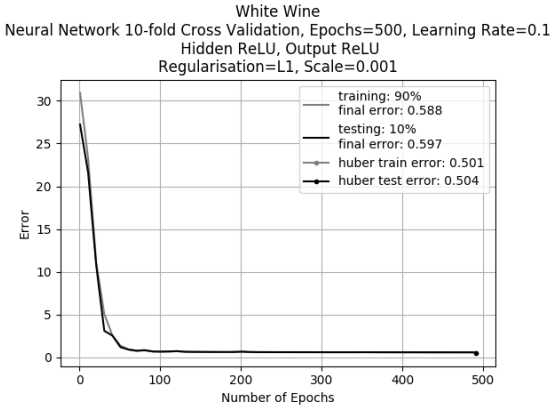

Figure 5. Best Neural Network White Wine predictor.

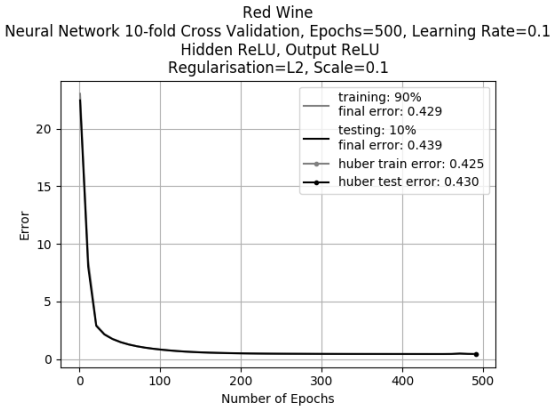

Figure 6. Best Neural Network Red Wine predictor

for an SVR with no regularisation was 0.189, however with L_1 regularisation with $\lambda = 0.1$ this was worsened to 0.272.

The algorithm was also tested with a learning rate of 0.05 and with 500 epochs, achieving a huber test error of also around 0.2.

6. Conclusion

Models were trained and tested on a large dataset of both white and red wines. The task was to obtain a regression predictor with small test error. It was found that a linear approach to this problem was successful. Multiple basic and advanced algorithms were implemented and tested, resulting in the conclusion that neural network models over fit the data and require regularisation, however even with this do not compete with linear methods less prone to overfitting.

Most linear regression methods had a similar success rate, SVR appeared to be very consistent.

It was found that the test error on the red wine data set was less than the white, this is due to the data being highly

	LASSO	Ridge	SVR	Huber
No Reg	0.264	0.260	0.271	0.256
L1 Reg	$\lambda = 0.0001$ 0.261	$\lambda = 0.001$ 0.268	$\lambda = 0.001$ 0.267	$\lambda = 0.01$ 0.265
L2 Reg	$\lambda = 0.0001$ 0.265	$\lambda = 0.1$ 0.261	$\lambda = 0.01$ 0.269	$\lambda = 0.1$ 0.260
Elastic Net Reg	$\alpha = 0.1$ $\lambda = 0.01$ 0.260	$\alpha = 0.01$ $\lambda = 0.3$ 0.257	$\alpha = 0.1$ $\lambda = 0.001$ 0.257	$\alpha = 0.5$ $\lambda = 0.001$ 0.263

Table 2. Comparison of white wine best linear regression results after 500 epochs of training with a learning rate of 0.05.

concentrated in the former, making the prediction task easier.

7. Appendix

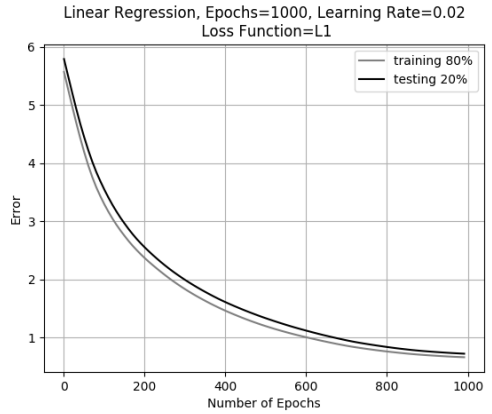

Figure 7. Loss Function = $\sum_{i=1}^{n} |y_i - f(x_i)|$, Training Error = 0.321, Testing Error = 0.373

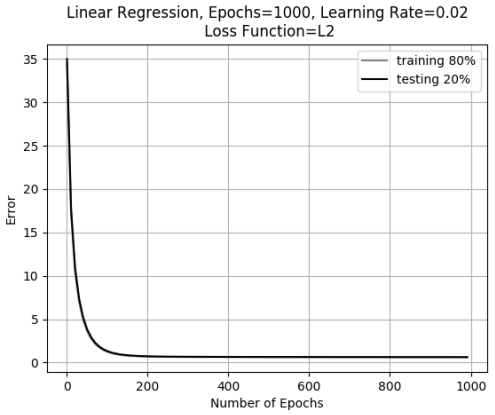

Figure 8. Loss Function = $\sum_{i=1}^{n} (y_i - f(x_i))^2$, Training Error = 0.271, Testing Error = 0.276

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`.