Classifying Wines based on Physicochemical Properties

Martin Ferianc
Imperial College London
mf2915@ic.ac.org, CID:00984924

1. Introduction

This work looked at a well-known problem targeting classification of Portuguese wines first presented by Cortez et al. in 2009 [1]. Red wines (*1599*) and white wines (*4898*) were classified into 7 classes (wine score ranging from 3 to 9) based on their quality and their 11 physicochemical features, giving *6497* wines-samples in total. This report presents an investigation of **logistic regression** and a **neural network** in pursuit of finding the highest accuracy, while classifying wines.

1.1. Previous Results

The work presented by Cortez et al. [1] considered two separate methods; this work is based on these and benchmarked with: a linear method - linear regression - and a more complex classifier - shallow neural network with one hidden layer. Previously, data was separated between red and white wines and the resultant accuracies with respect to the methods were as follows:

Table 1. Previous accuracies

	Red wine	White wine
Linear regression (%)	31.2 ± 0.2	25.6 ± 0.1
Neural network (%)	31.1 ± 0.7	26.5 ± 0.3

1.2. Data Preprocessing

It was deemed that by combining the two different wine types together, and keeping the type as a new feature, the final classifier would be more resistant towards over-fitting and it would also increase the amount of total available data. However, there was a risk of shaping the classifiers towards average white wines, as this type was by far the majority, see Figure 1. Each of the features had an implicitly different scale[1] and needed to be normalised. Later, to get a robust estimates of the average classification rate (CR), the data was split into 10 folds with 10% for *test*, 10% for *validation* and 80% for *training* set, respectively.

[1]The statistical details per feature, and correlation matrix between the features can be found in the Appendix A.

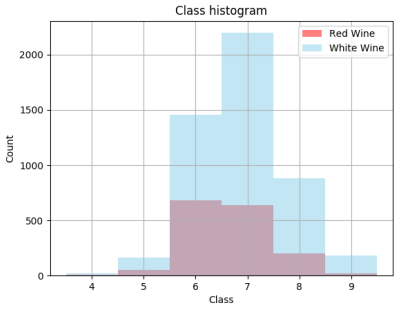

Figure 1. Histogram representation of number of samples in each class according to the type of wine

2. Logistic Regression Classifier

2.1. Theoretical Foundation and Definition of Error Measure

Logistic regression classifier is a method that is widely applied in practice [2] in classification problems. It is a classifier whose output $h(x)$ with relaxed features x can be interpreted as a probability for a binary event. Where $h(x)$ is defined as:

$$h(x) = \theta(w^T x)$$

where w^T is a row of learnable weights, x is the input vector and $\theta(x)$ is the logistic function $\frac{e^x}{1+e^x}$. It is a suitable baseline method for the evaluated problem as its output is simple to interpret in terms of probability of a discrete output. However, it generalizes the data into linearly separable regions, which is not true in practical considerations and is considered one of the main limitations of this algorithm.

The standard error measure used in logistic regression is based on the notion of likelihood. Where through maximum likelihood estimator we can define the error measure $E(w)$ that is attempted to be minimized in terms of w, feature vector x_i, labels y_i and the total number of data N as:

$$E(w) = \frac{1}{N}\sum_{i=1}^{N} ln(\frac{1}{\theta(y_i w^T x_i)}) = \frac{1}{N}\sum_{i=1}^{N} ln(1 + e^{y_i w^T x_i}) \quad (1)$$

Unfortunately, unlike the case of linear regression, the mathematical problem is non-convex and the form of the gradient of $\nabla E(w)$ is not straightforward to manipulate and an analytic solution is not feasible. Therefore, the best approach would be to do so iteratively, through gradient descent to approximate it as close to zero as possible.

This algorithm is computationally expensive, as it is necessary to traverse through all the training examples to compute the gradient. This work considered mini-batch gradient descent, that is more effective but overall less accurate, in comparison to normal gradient descent.

2.2. Implementation Details

By definition the logistic regression classifier is a binary classifier. To work around this, each classifier was trained in classification of only one score against all the others, in the standard *one vs. all* (OvA) fashion and combined with 6 other classifiers trained, to classify the wines with one concrete score. *Scikit-learn* [3] library enabled this setup with their class *SGDClassifier*.

To find the highest test/validation accuracy, there is a number of hyper-parameters that can be varied while training the classifier. This work looked at optimizing the:

- *Learning rate* η, to achieve the global minimum as fast and as accurately as possible.
- *Number of epochs* required for training until diminishing returns were observed.
- *Regulariser* and *regularisation penalty* to avoid over-fitting and improve the accuracy on validation and test data.

2.2.1 Learning Rate

It was important to investigate an optimal learning rate that droved the weights towards a global minimum. It was expected that the ideal loss function would degrade quickly in the first n iterations and stabilize later on. The tests were performed with learning rates: $0.1, 0.01, 0.001, 0.0001$ over 5 epochs with a batch size of 256, without a regulariser defined.

Table 2. Learning rate affecting validation accuracy

Learning Rate	0.1	0.01	0.001	0.0001
Val. CR (%)	36.4	43.0	46.2	46.7

Note that in the Figure 2, each line represents the average loss at that iteration over 10 folds. It can be seen from the data that the best learning rate in terms of loss is 0.001, however not in terms of validation accuracy as seen in Table 2. Despite this fact, it was used in later tests at it exhibits the smallest variance in the validation accuracy.[2]

[2]The plot can be found in Appendix B.

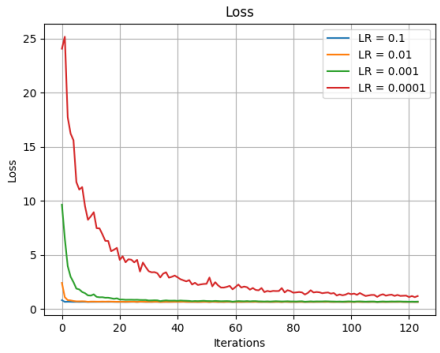

Figure 2. Loss function per different learning rates

2.2.2 Number of Epochs

With mini-batch gradient descent we can vary the number of times the classifier sees all the data and again it was important to find the right number to achieve the global minimum in the loss function but also prevent over-fitting to the training data. The tests were performed with $1, 3, 5, 7, 9$ epochs.

Table 3. Number of epochs affecting validation accuracy

Epochs	1	3	5	7	9
Val. CR (%)	44.3	45.5	46.1	46.3	46.5

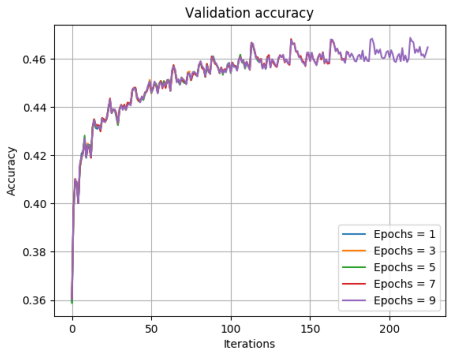

Figure 3. Validation accuracy per total number of iterations

Based on the results in Figure 3 and Table 3 it can be seen that the ideal number of epochs remained 5, with the batch size of 256 samples. Increase in epochs did not bring a significant improvement in accuracy, that would justify the trade-off in computation needed to perform more optimisations.

2.2.3 Regularisation

To avoid over-fitting to noise in the data elastic net has been considered as a viable regularisation method. Its advantages

are, that it includes both L1 and L2 regularisers as special cases, and beneficially with respect to the wine problem, that it is resistant towards correlated features [4]. The new loss function then becomes:

$$E_{Reg}(w) = E(w) + \lambda_1 \sum_{k=1}^{K} |w_k| + \lambda_2 \sum_{k=1}^{K} w_k^2 \quad (2)$$

Where w_k are the weights for the individual features, λ_1, λ_2 are the regularising penalties and $E(w)$ is the previous loss. Both can be expressed in a single relationship as:

$$\alpha = \frac{\lambda_2}{\lambda_1 + \lambda_2} \quad (3)$$

By varying α and keeping λ_1 at 0.15 we achieved different effects on the validation accuracy.

Table 4. Elastic net regularisation and validation accuracy

α	0.1	0.01	0.001	0.0001	0.00001
Val. CR (%)	44.8	45.9	46.0	46.2	46.5

As seen in Table 4, the best logistic regression classifier was trained in 5 epochs with a batch size of 256 and a learning rate of 0.001, α set as 0.00001, λ_1 as 0.15 with test accuracy of 46.6$\pm$ 0.1% with 95% confidence.[3]

3. Neural Network

3.1. Theoretical Foundation and Definition of Error Measure

The main advantage of artificial neural networks (ANNs) is the ability to adapt to complex non-linear regions. They were proven to achieve high accuracy in classification challenges [5] in similar problems and hence were deemed as appropriate classifier for this problem.

For the complexity of the data set and the number of features, it is assumed that a feed-forward neural network with only a few fully connected layers was sufficient. Based on the previous research [1], more complex models might be more prone to over-fit to the data and hence were deemed as unsuitable.

A feed-forward ANN is defined in terms of an input layer, hidden layers and an output layer. Each layer consists of neurons with activation function $\theta(x)$ that have connections to neurons in the previous and next layer. The neuron sums inputs x_{ij} from previous layer i with applied weights w_{ij} and applies an activation $\theta(x)_{jk}$ that produces output for the next layer j, o_{jk} .

$$o_{jk} = \theta_{jk}(\sum_{j=1}^{J} w_{ij} x_{ij}) \quad (4)$$

[3]Confusion matrix and scores per class can be found in Appendix B and A.

Training is performed by iteratively applying the backpropagation algorithm [6] to find the best accuracy. For the purposes of classification, a widely used loss function $E(w)$ is categorical cross-entropy [7] with labels y_i, predictions o_i with respect to N data.

$$E(w) = -\sum_{i=1}^{N} y_i \ln o_i \quad (5)$$

This loss is often related to the last layer, that is interpreted as probability distribution upon the classes, whose activation function is softmax [8].

3.2. Implementation Details

This work made use of *Tensorflow* [9] library. The labels of the data had to be changed into one-hot encoding to be compatible with the softmax function in the last fully-connected layer. Again, as for the logistic regression classifier there are a number of parameters that can be varied to achieve the highest accuracy:

- *Number of neurons*, to better separate non-linear regions between each other.
- *Number of hidden layers*, multi-layer network should be capable of expressing a larger variety of non-linear surfaces simultaneously.
- *Activation function*, different activation functions may affect the convergence of the training process and also the final accuracy.

3.2.1 Number of Neurons

Based on the 12 features the complexity of the classifier and the number of neurons was not expected to reach a point beyond 100 neurons where the capacity of the network is too complex and expected to over-fit to training data.

Hence the exploration was inspired from the previous research [1] and began with 12-7-7 (12 neurons in the input layer, 7 neurons in the hidden layer and 7 neurons in the output layer) model with $7, 10, 20, 40, 80, 100$ neurons in the hidden layer with learning rate 0.001, 5 epochs for training and batch size 256 samples, elastic net regularisation and the base activation function in the hidden layer was ReLU where $\theta(x) = max(0, \sum_{j=1}^{J} w_{ij} x_{ij})$. The update rule was pre-optimized Adam. Dropout rate was set to 0.7. Dropout was used to force the neurons to be self reliant in making decisions and to develop their own spatial distinct features. Note that after each fully-connected layer the weights in the individual layers, except for the input neurons, were batch normalised.

Table 5. Number of neurons affecting validation accuracy

Neurons	7	10	20	40	80	100
Val. CR (%)	49.0	50.5	51.2	51.5	52.9	51.3

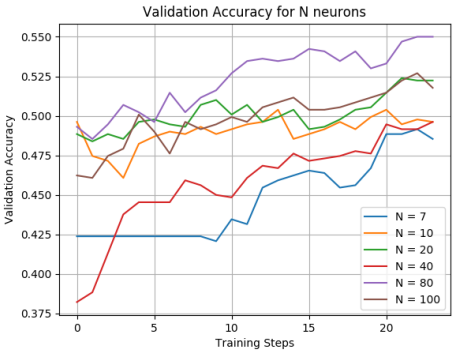

Figure 4. Validation accuracy of networks with different number of neurons with respect to the number of training steps

Empirically, from Figure 4 and Table 5 it was determined that 80 neurons had the best validation accuracy and so this model was used for further optimizations. The interpretation of this can be explained in neurons creating several non-linear separating regions for corresponding wine quality inside the same layer. Note, that with 100 neurons the model had significantly better training accuracy but smaller validation accuracy, which can be accounted to over-fitting to the training data.[4]

3.2.2 Number of Hidden Layers

By increasing the number of hidden layers, the model should be able to better generalize the separation of individual scores into non-linear regions. Therefore, using the result from the previous section 4 different models were tested with an increasing number of hidden layers.

Table 6. Number of hidden layers affecting validation accuracy

Hidden layers	1	2	3	4
Val. CR (%)	55.4	55.0	53.4	51.9

Interestingly, the validation error as seen in Table 6 increased with an increased number of hidden layers in comparison to the training error. This is accounted to unnecessarily over-fitting to the training data, given the total number of features per sample is only 12 features and hence by the principle of Occam's razor [10] we continued with the simplest hypothesis, with the best validation accuracy.

3.2.3 Activation Function

It was found that picking the right activation function can greatly accelerate the convergence of stochastic gradient descent and in the end help to achieve better testing or validation accuracy [11]. The basis activation function ReLU, which is computationally inexpensive, has inherent disadvantages. It completely drops negative values and that it can saturate its neuron with large gradients and then the next gradient flowing through the unit will be zero for the remainder of the training [12]. Hence, it was deemed to investigate other non-linearities that have saturating outputs such as popular tanh $\theta(x) = \tanh(x)$ and elu:

$$\theta(x) = \begin{cases} x, & \text{if } x \geq 0 \\ e^x - 1, & \text{otherwise} \end{cases}$$

Table 7. Activation function affecting validation accuracy

Activation Function	ReLU	tanh	elu
Val. CR (%)	55.6	50.5	53.9

Despite its disadvantages ReLU, was proven to give the best performance in this model as seen in Table 7. The best architecture found in this exploration was 1-80-7 model with ReLU as activation function with dropout rate 0.7 and elastic net regulariser trained in 5 epochs with the batch size 256. The test accuracy was $53.0 \pm 0.1\%$ with 95% confidence.[5]

4. Conclusion

Overall, this exploration showed that logistic regression achieved better combined test accuracy in solving this problem than linear regression classifier presented originally by Cortez et al. by 13.3% [1]. Furthermore a denser neural network, with more neurons than the original implementation, improved on the test accuracy in average by 24.7%.

The biggest limiting factor in this problem is contributed to the dataset being imbalanced, both in terms of classes and wine types, where the average labels skewed the classifiers towards these scores. Both of these problems could be solved by collecting more samples per different classes and wine types that would guarantee that the dataset is balanced and the classifier unbiased.

Future work that could not be completed in this scope of investigation, includes combining the best models in each algorithm class with optimal parameters into an ensemble of multiple classifiers each trained on different features and samples of the training data. This approach would greatly reduce the variance of the hypothesis space and probably achieve higher testing classification rate.

In addition, to choosing different classifier combinations, further investigation of this problem could be done in the realm of transforming the features and their combinations into higher dimensions, that would uncover unseen relationships in the data. Lastly, the hyper-parameters could have been explored exhaustively to find the most optimal model, which would require immense computational power.

[4]Loss function for this model can be found in Appendix C.

[5]Confusion matrix and scores per class can be found in Appendix C.

Pledge

I, Martin Ferianc, pledge that this assignment is completely my own work, and that I did not take, borrow or steal work from any other person, and that I did not allow any other person to use, have, borrow or steal portions of my work. I understand that if I violate this honesty pledge, I am subject to disciplinary action pursuant to the appropriate sections of Imperial College London.

I have discussed my work with Alex Montgomerie-Corcoran and Rajan Patel.

Source code

All code and instructions to reproduce the results can be found at `https://github.com/fexter-svk/WineClassification-EIE3`.

References

[1] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, 1998.

[2] .-a. Abu-Mostafa, Yaser S., *Learning from data : a short course*. United States]: AMLBook.com, 2012.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, pp. 301–320, April 2005.

[5] T. M. Mitchell, *Machine Learning*. WCB McGraw-Hill, 1997.

[6] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015. Published online 2014; based on TR arXiv:1404.7828 [cs.NE].

[7] C. M. Bishop, *Pattern recognition and machine learning*. Information science and statistics, New York: Springer, 2006.

[8] K. Duan, S. S. Keerthi, W. Chu, S. K. Shevade, and A. N. Poo, "Multi-category classification by soft-max combination of binary classifiers," in *Multiple Classifier Systems* (T. Windeatt and F. Roli, eds.), (Berlin, Heidelberg), pp. 125–134, Springer Berlin Heidelberg, 2003.

[9] 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.

[10] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics, New York, NY, USA: Springer New York Inc., 2001.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[12] B. Wang, T. Li, Y. Huang, H. Luo, D. Guo, and S. J. Horng, "Diverse activation functions in deep learning," in *2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, pp. 1–6, Nov 2017.

A. Data Analysis

Table 8. Analysis of the data

Feature name and no.	mean	std	min	max
0. Wine Type	0.75	0.43	0.00	1.00
1. Fixed Acidity	7.22	1.30	3.80	15.90
2. Volatile Acidity	0.34	0.16	0.08	1.58
3. Citric Acid	0.32	0.15	0.00	1.66
4. Residual Sugar	5.44	4.76	0.60	65.80
5. Chlorides	0.06	0.04	0.01	0.61
6. Free Sulfur Dioxide	30.53	17.75	1.00	289.00
7. Total Sulfur Dioxide	115.74	56.52	6.00	440.00
8. Density	0.99	0.00	0.99	1.04
9. pH	3.22	0.16	2.72	4.01
10. Sulphates	0.53	0.15	0.22	2.00
11. Alcohol	10.49	1.19	8.00	14.90

It can be seen that each of the values had a different scale and needed to be normalised accordingly. Note that the features were normalised with respect to the training set and not all data.

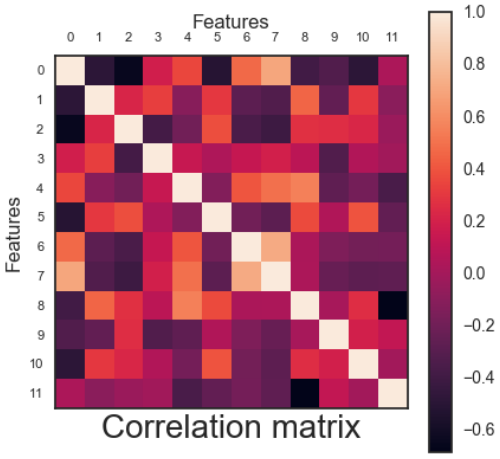

Figure 5. Correlation matrix.

Many of the features are correlated and in particular features 7 and 0 have shown the highest correlation.

B. Logistic Regression

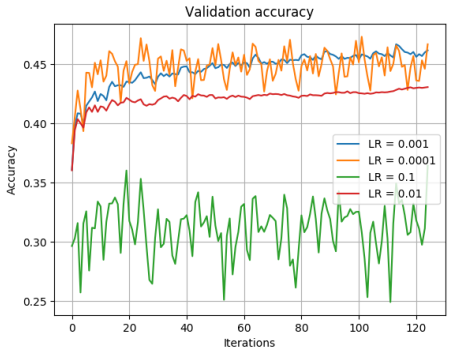

Figure 6. Validation accuracy of different learning rates.

It can be seen that the learning rate 0.001 achieved the least variance in the validation accuracy over the number of iterations and represents a more stable classifier that could be used in practice.

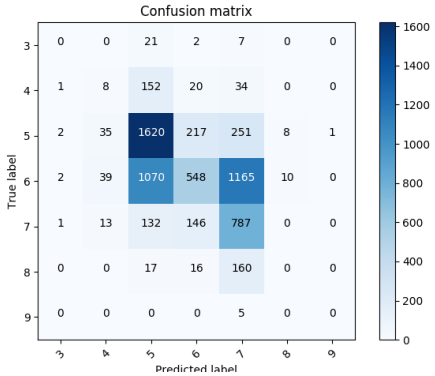

Figure 7. Confusion matrix for the linear method

Table 9. Statistical Scores

Class	3	4	5	6	7	8	9
F1 measure	0	10	65	26	41	0	0
Recall (%)	0	11	76	17	67	0	0
Precision (%)	0	10	57	54	41	0	0

Note that it can be seen that the classifier is largely biased towards the average classes due to the overwhelming amount of samples in these categories, giving low F_1 measure for all other categories respectively.

C. Neural Network

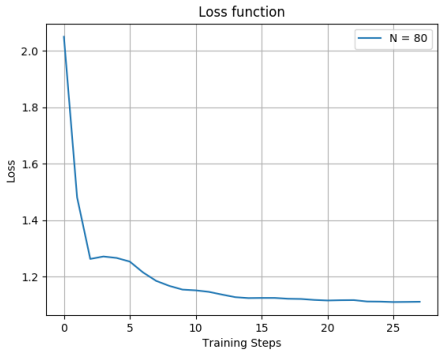

Figure 8. Loss function for the optimal neural network

It can be seen that it is unnecessary to train beyond 5 epochs as the loss function as well as validation error did not decrease further.

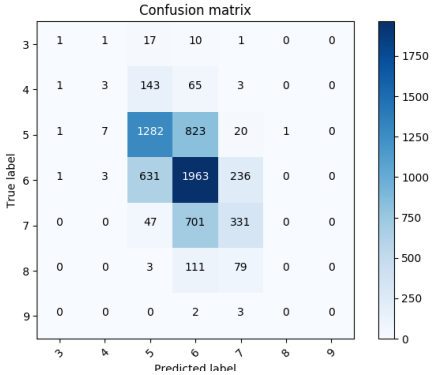

Figure 9. Confusion matrix for the neural network

Table 10. Statistical Scores

Class	3	4	5	6	7	8	9
F1 measure	15	11	62	51	32	0	0
Recall (%)	10	6	62	71	24	0	0
Precision (%)	30	48	61	54	48	0	0

Note that also neural network classifier showed a bias towards the average classes 5-6, because of the data being completely imbalanced as shown in Figure 1.