Machine Learning Project: Wine Quality Prediction

Yao Lei Xu
Department of Electrical and Electronic Engineering
Imperial College London
SW7 2AZ
yao.xu15@imperial.ac.uk

I, Yao Lei Xu, 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.

1. Abstract

This report explores the performance of different machine learning models when predicting wine quality from their physicochemical composition. Several linear and non-linear methods are implemented using MATLAB, with Gaussian kernel based support vector machine obtaining the best results.

2. Analysis

2.1. Introduction to the problem (part A)

The original dataset [1] is composed of 12 attributes: 11 physicochemical features and 1 quality score ranging from 0 to 10. Although the red wine and white wine data were initially separated, for this project, they are merged together with an additional binary feature distinguishing their colour, making a total of 12 features that can be used for prediction. The figure below shows the distribution of the wine quality for the complete dataset:

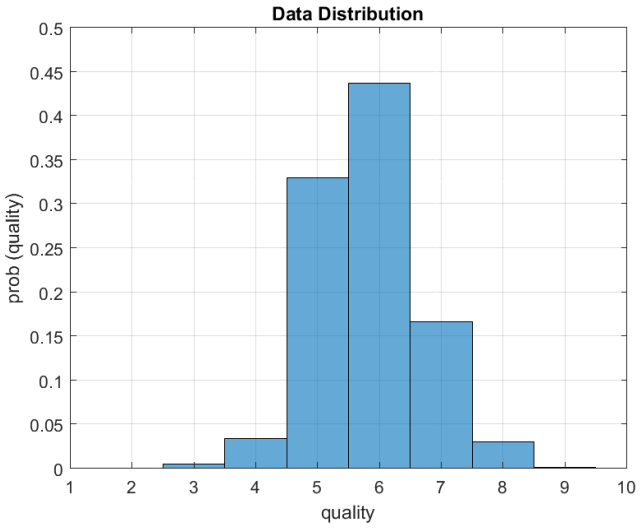

Figure 1 – Data distribution

It is essential to have a training set that resembles the distribution of the overall data in order to train a good machine learning model, hence producing predictions that follow a similar distribution. Therefore, the samples are always randomly shuffled and chosen with uniform probability when split into training and test sets.

Mathematically, the dataset consists of a 6497x12 input matrix X and a 6497x1 output matrix Y (12 features and 6497 samples total), where 70% of the samples (4547 randomly chosen samples) will be used for training, and the remaining 30% (1950 samples) for testing. Once splitting the data, to speed up the convergence of various algorithms, all columns of the training set X_{tr} are normalised to zero mean and unit variance, and the columns of the test set X_{te} are standardised using the mean and standard deviation from X_{tr} (It is essential to perform standardisation only after the split to prevent data snooping). The performance of various machine learning models will be assessed using the test error:

$$R(h) = E\left[l\left(h(x), f(x)\right)\right] \quad (1)$$

where $h(x)$ is the hypothesis, $f(x)$ the target function and $l\left(h(x), f(x)\right)$ the loss function (note that the training and test error mentioned in the report are always averaged over 30 repetitions with different sets of X_{tr} and X_{te}).

For the loss function, it is possible to use the binary error $I(h(x) \neq f(x))$ to assess the probability of estimating the quality correctly, but this is not the optimal approach. For instance, consider a hypothesis h_1 which can predict 50% of the data correctly while missing the other 50% by a big range (e.g. $h(x_i) = 1$ while $f(x_i) = 10$). Consider another hypothesis h_2 for which all predicted values are wrong but with an error of just 1 (e.g. $h(x_j) = 6, f(x_j) = 7$). The latter has $E\left[l\left(h_2(x), f(x)\right)\right] = 100\%$, which is twice as much compared to h_1, even if the performance is arguably better. Therefore, considering the above argument and that there are no particular reasons to prefer over or under estimation of wine quality, the loss function chosen for this problem is the squared error. Hence, the test error (mean squared error) is defined as:

$$R(h) = E\left[\left(h(x) - f(x)\right)^2\right] = MSE \quad (2)$$

Lastly, it is possible to treat the problem as a multi-class classification problem, but since there are little or no samples available for wine of very high or very low qualities, it would be hard to generalise for those values. Hence, a regression-based approach will be used to produces predictions that will be rounded to the nearest integer (this however significantly increases the mean squared error compared to non-rounded values) and capped in the range from 0 to 10 (although the models never produced predictions outside of this range).

2.2. The baseline predictor (part B)

Linear regression with Tikhonov regularisation (Ridge regression) will be used as the baseline predictor. The weights are defined as [2]:

$$w_{reg} = (X^T X + \lambda \mathrm{I})^{-1} X^T y = V\, diag\left(\frac{d_i}{{d_i}^2 + \lambda}\right) U^T y \quad (3)$$

Where λ is the parameter to be optimised via cross validation (cv).

Ridge regression is preferred in this case over a simple linear model for 2 main reasons. Firstly, it reduces the overall overfitting by increasing λ. Secondly, from a stability point of view, w_{reg} is better than its linear counterpart ($\lambda = 0$), since d_i approaches 0 if X^TX is singular (or nearly), causing linear regression to blow up. In Ridge regression however, the presence of λ avoids this issue, making it a better choice despite being a biased estimator.

The ridge regression will be carried out using the MATLAB function 'fitrlinear' with stochastic gradient descent as the objective function minimisation algorithm. This function returns a Ridge regression model for a given parameter of λ which can be used for the test set as well.

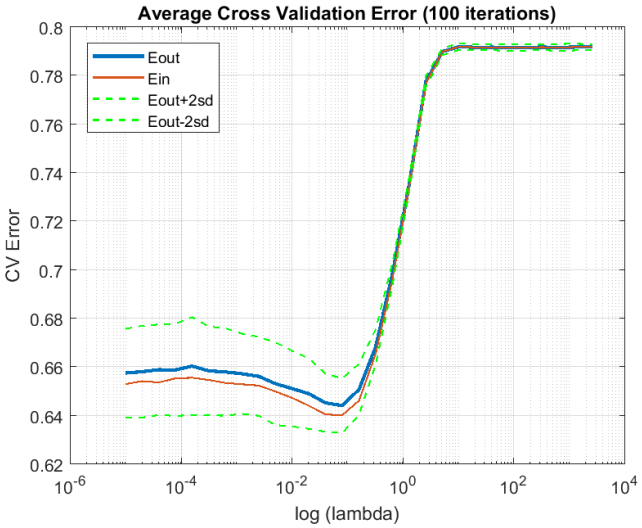

Figure 2 – Ridge cross validation

The figure above plots the 10-fold cross validation error Eout averaged over 100 repetitions (the blue line) for 30 different values of $\lambda \in [0, 2700)$ (see the MATLAB code for the exact values). The green dotted lines bound Eout within 2 standard deviations, while the red line is the cv training error. Every time, the dataset X_{tr} is split into different D_{train} and D_{val} for each iteration of the 10-fold cv by randomly selecting the samples, which are again normalised with the cv training set parameters.

As expected, Eout grows as lambda increases to infinity since it is increasingly under-fitting. However, there is a minimum at λ = 0.041 with a corresponding Eout of 0.642, which is slightly better than the simple linear regression ($\lambda = 0$, Eout = 0.659). The entire training set is then used to produce the final hypothesis g_m using the optimal λ of 0.041, which produced an average test error of 0.640. Also, it is important to notice that for small values of λ (as it approaches the linear regression model), Eout has greater variance in performance due to more overfitting.

2.3. Advanced machine learning methods (part C)

For more advanced methods, non-linear models will be considered. Performance wise, non-linear models are expected to perform better than the linear ones in terms of in-sample error as they are better at fitting the data, but the test error is likely to be worse. For instance, given the target function $f(x)$, the bias-variance trade-off is as follows [2]:

$$E_{D,\epsilon}\left[(g^{(D)}(x) - y)^2\right] = var + bias \quad (4)$$

$$var = E_D\left[(g^{(D)}(x) - \bar{g}(x))^2\right] \quad (5)$$

$$bias = E_D\left[\left(\bar{g}(x) - f(x)\right)^2\right] \quad (6)$$

$$\bar{g}(x) = E_D\left[g^{(D)}(x)\right] = average\ hypothesis \quad (7)$$

Hence, a more complex model has a larger hypothesis class H which allows the $\bar{g}(x)$ to approximate the target function $f(x)$ better (smaller bias). However, a larger hypothesis class means that $g^{(D)}(x)$ will vary more according to the dataset provided (higher variance).

A possible solution to this issue is the support vector machine (SVM) with non-linear kernels, as the SVM applies its model to a larger non-linear features space, but it is not penalised by its high dimensionality. For this reason, advanced machine learning methods such as SVM with polynomial kernel (equation 8) and with Radial Basis Function (RBF or Gaussian) kernel (equation 9) will be explored for this section.

$$k(x_1, x_2) = (1 + x_1{}'x_2)^p \quad (8)$$

$$k(x_1, x_2) = e^{(-\gamma||x_1 - x_2||^2)} \quad (9)$$

The polynomial kernel is more computationally efficient compared to RBF since for a given input x of size d and a polynomial order q, there are only d^q terms to work with. The RBF on the other hand is less efficient, since the kernel maps to an infinite dimensional space due to its exponential term. However, performance wise, it allows the SVM to construct its model in this massive space and map it back, making it a great universal function approximator.

2.4. Implementation of Advanced Methods (part D)

For the implementation of the SVM methods, the MATLAB function 'fitrsvm' is used with the following specifications:

KernelFunction	rbf	polynomial
KernelScale	1/(sqrt(gamma))	-
PolynomialOrder	-	Order
Solver	L1QP	L1QP
CacheSize	Maximal	Maximal
Epsilon	iqr(y)/1.349	iqr(y)/1.349
BoxConstraint	iqr(y)/1.349	iqr(y)/1.349

Table 1 – SVM specifications

The 'KernelFunction' sets the kernel that needs to be implemented. The 'KernelScale' implements the value of γ in equation 9 for RBF, while the 'PolynomialOrder' sets p in equation 8 for the polynomial kernel. The 'Solver' L1QP performs optimisation via L1 soft-margin quadratic programming that minimises $||w||^2 + c\sum_{i=1}^{n}\varepsilon_i$ subject to $y_i(w^Tx_i + b) \geq 1 - \varepsilon_i$ and $\varepsilon_i \geq 0$. Here, $\sum_{i=1}^{n}\varepsilon_i$ is the total margin violation and c is the box constraint, which controls the penalty of observations that are outside of epsilon margin, which can reduce overfitting [3]. The 'CacheSize' is just to reduce the computation time by reserving more memory for its calculations. Lastly, 'Epsilon' and 'BoxConstraint' are set to $\frac{iqr(y)}{1.349}$ (default MATLAB value for the RBF kernel), which are around $\frac{1}{10}$ of the standard deviation of the training set wine

quality [4]. Note that although 'Epsilon' and 'BoxConstraint' are set to default values recommended by MATLAB, it would be best to optimise them for better performing models, which is done in section 2.5.

As before, X_{tr} is split randomly into smaller D_{train} and D_{val} for the k-fold cross validation method, where the data are normalised again. The cv error $\hat{R}_{CV,K} = \frac{1}{10}\sum_{k=1}^{10} \hat{R}_{val,k}(g_k^-)$ generated from the models specified above are then compared for different hypotheses (different values of γ and p). Finally, a model is trained using the hypothesis for which $\hat{R}_{CV,K}$ is minimal and the final training and test error are computed. Note that to reduce the computational complexity for these advanced models, the 5-fold cross validation is used instead.

2.5. Performance of Advanced Methods (part E)

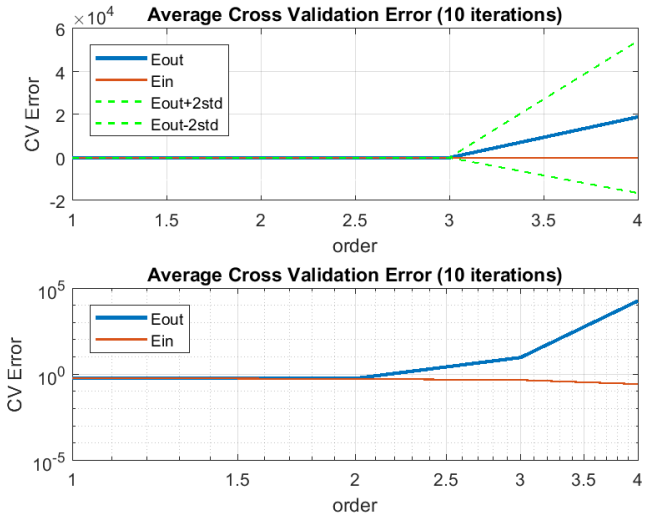

Figure 3 – Polynomial based SVM cross validation
Regular and log-log plot

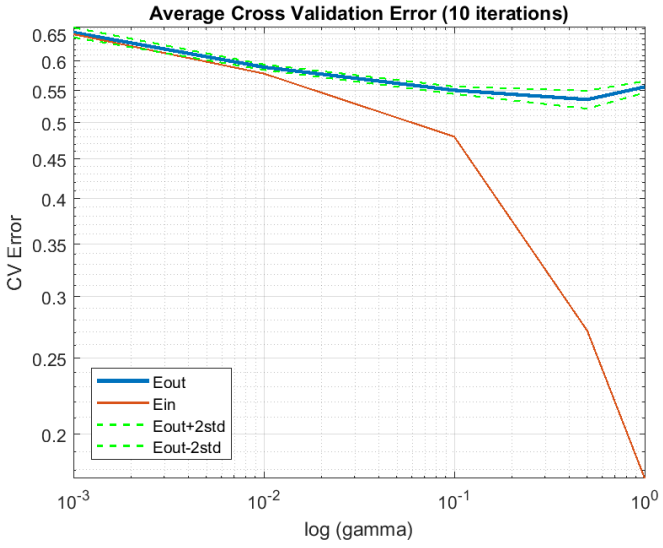

Figure 4 – RBF based SVM cross validation with γ

Figure 3 and 4 show the performance of the polynomial and RBF model for different parameters. As expected, the CV training error decreases as a function of the model complexity (increasing γ or p) reaching almost zero error, while the validation error $\hat{R}_{CV,K}$ (Eout) and its variance increases due to overfitting and increasing complexity. After determining the best values of γ and p using cross validation, a final model is trained and used to compute the training and test error.

The following table summarises the performance of all models considered so far, comparing their mean square error:

	Min cv error	Training error	Test error	Best parameter
Ridge regression	0.642	0.637	0.640	λ = 0.041
SVM (Polynomial)	0.597	0.574	0.602	p = 2
SVM (RBF)	0.536	0.275	0.514	γ = 0.500

Table 2 – Performance Comparison

From the table, the best performing model is the support vector machine with Gaussian kernel, which has both the smallest training error and the smallest test error, while SVM with polynomial kernel is the second best due to its slightly higher complexity compared to linear. This is expected for the same reasons outlined in section 2.4.

Furthermore, although the 5-fold cv error used for SVM is a worse estimate of the test error compared to the 10-fold, the percentage error in estimation from the above data is only 3% on average. In addition, smaller cv error corresponds to smaller test error overall, which confirms that the 5-fold cv method is an acceptable choice.

Finally, the RBF based SVM is then further optimised by cross validating the BoxConstraint value (which helps with regularisation), producing the best solution reached by this project (note that this is a locally optimal model, not global. Ideally, it needs to be optimised with respect to all variables, but due to time constraints it is left as future work. See appendix 4.1 for the unfinished attempt). The characteristics of the final model are as follows:

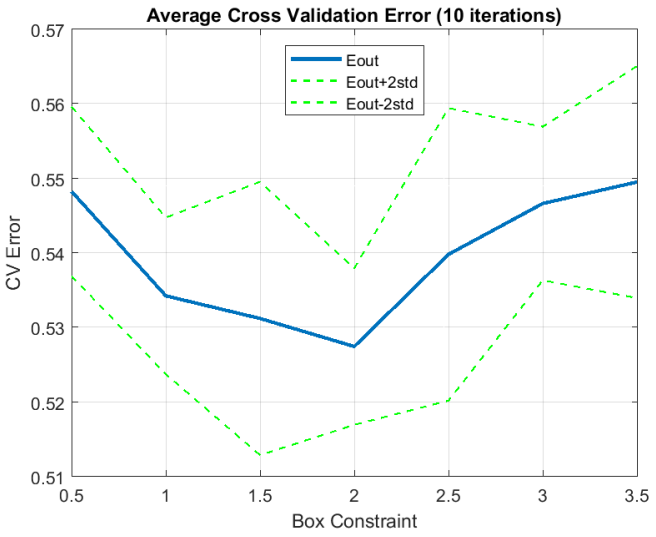

Figure 5 – RBF based SVM cross validation with box constraint

Min CV Error	Training Error	Test Error	Box constraint	Gamma
0.5275	0.143	0.512	2	0.5

Table 3 – Final RBF based SVM characteristics

2.6. Conclusion (part F)

One of the core principles in machine learning is about fitting the data in the best way possible while considering the penalty introduced by using complex hypothesis classes. For this reason, it comes with no surprise that the SVM with Gaussian kernel was the best performing model in terms of mean squared test error. For instance, the kernel maps to an infinite dimensional space to work on, making it a universal function approximator, which is then further optimised by selecting γ through cross validation. Finally, its complexity is penalised through the choice of a suitable box constraint.

The trained machine is now able to judge the quality of the wine with a mean squared error around 0.5, guessing the exact value of the wine 63% of the times, which is better than most humans (including the author of this report). However, whether the machine has learned to appreciate a good glass of wine and is able to act as a professional sommelier is left for debate. For instance, machine learning is about producing the most probable value given a data set, and since most of the wine were of quality 5 or 6, the best and most common guesses produced by the model are 5's and 6's as well.

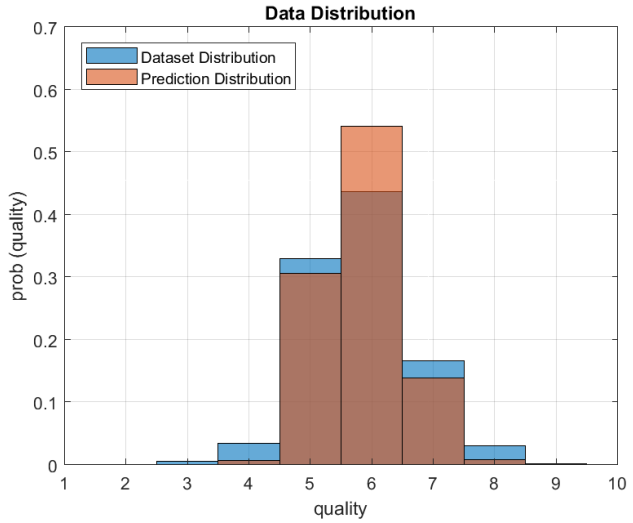

Figure 6 – Prediction Distribution

Finally, not only sommeliers often consider features such as the colour and the smell of the wine to judge the quality (which are not present in the data), the opinions of wine experts are also known for varying greatly for the same wine. Therefore, a final mean square test error of 0.5 is more than acceptable, since this means the machine predictions are often at most 1 unit away from the target value. For instance, if constraints are relaxed such that the predictions are considered correct if $|h(x) - f(x)| \leq 1$, it would predict correctly 96% of the time.

3. References

[1] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. (2009). *Decision Support Systems: Modelling wine preferences by data mining from physicochemical properties.* [online] Available at: https://www.sciencedirect.com/science/article/pii/S0167923609001377?via%3Dihub [Accessed 19 Mar. 2018].

[2] A. György, K. Mikolajczyk, D. Gündüz. (2018*). Imperial College London, EE3-23 Machine learning: lecture notes*

[3] Mathworks. (2018). *Understanding Support Vector Machine Regression.* [online] Available at: https://uk.mathworks.com/help/stats/understanding-support-vector-machine-regression.html [Accessed 21 Mar. 2018].

[4] Mathworks. (2018). *Fit a support vector machine regression model* [online] Available at: https://uk.mathworks.com/help/stats/fitrsvm.html [Accessed 21 Mar. 2018].

4. Appendix

4.1. Optimisation of RBF based SVM

Due to time constraints the final solution presented here is only a locally optimal model, not global. For better performance, the error must be optimised with respect to all parameters. Below is a figure plotting the cv error with respect to gamma and box constraint, which suggests the possibility of a better performing model, where the cv error is 0.479 for gamma = 0.1 and box constraint = 4.5.

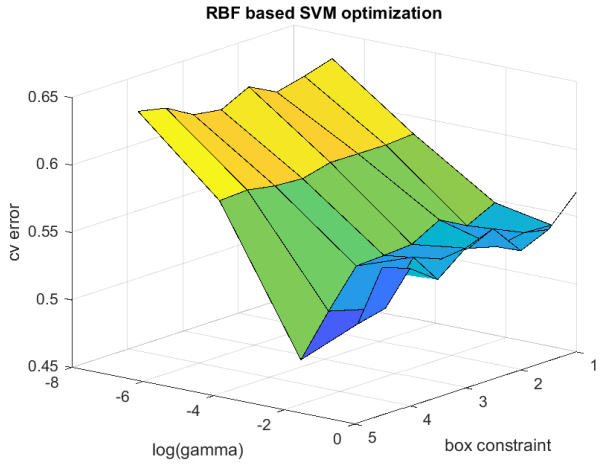

Figure 7 – Optimisation of RBF based SVM