Machine Learning Project 2017: Predicting Wine Quality by Analysing Physiochemical Attributes

Rajan Patel
CID: 01062228
Imperial College London
rnp15@ic.ac.uk

I. INTRODUCTION

A. *Problem Overview*

The Global Wine Industry was valued at $287.39 billion in 2016 and is expected to reach $402 billion by 2022 at an CAGR (Compound Annual Growth rate) of 5.8%[1]. In order for wine producers to capture this growth and expand their market share in this industry, they tend to promote their products using wine quality certifications. Obtaining a certification is a long and expensive ordeal as the quality of wine is assessed by multiple human wine experts that belong to a particular standards body such as the UKVA (United Kingdom Vineyards Association)[2].In order to streamline, speed up and cheapen this process, the wine industry is exploring new technologies such as machine learning to see how accurately they can assess the quality of wine.

In this report, the Wine Quality Data Set from the UCI Machine Learning Repository[3][4] has been used to explore the effectiveness of linear models such as Linear Regression and more complex models such as Neural Networks, in determining the quality of wine.[1]

B. *Dataset Exploration*

The Wine Quality Data Set is originally separated as Red and White Wine Data Sets that are variants of the Portuguese ”Vinho Verde” wine, which have 1599 and 4898 data points respectively. Each data point has 11 physiochemical input features consisting of; fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates and alcohol. Each data point has 1 discrete real-valued output feature of Wine Quality ranging from 0 (Worst quality) to 10 (Best quality). The wine quality rating is based on sensory data calculated from the median of at least 3 evaluations made by wine experts. By combining the two data sets, 6497 data points are obtained and another feature can be added which is the ’Type of Wine’.[2]

C. *Preprocessing*

Following from the feature analysis in Appendix A, it was revealed that the type of wine as a feature has very little correlation with the quality, hence it can be excluded as a feature and thus acceptable to consider the combined data set for training and testing models.

In the Red Wine Data Set (RWD), White Wine Data Set (WWD) and Combined Data Set (CDS), total sulfur dioxide and free sulfur dioxide had a high correlation. Free sulfur dioxide can be excluded from all data sets as a feature, since it had little correlation with quality. In the CDS and RWD, residual sugar can be excluded as it had no correlation with quality. In the WWD, residual sugar had a high correlation with density, but a very low correlation with quality, hence was excluded as a feature. In Section II, the performance of all 3 data sets will be compared with 9 input features (free sulfur dioxide and residual sugar excluded).

If we considered the data sets on a purely individual basis (not compared with each other), other features could be excluded. For example, in the RWD fixed acidity had a high correlation with citric acid, pH and density, but very low correlation with quality, hence could be excluded as a feature. In the WWD, citric acid had a negligible correlation with quality, thus could be excluded as a feature. In all data sets Alcohol and volatile acid usually had the highest correlation with quality, thus an analysis of these two features exclusively could have been carried out, but this has been done before[5].

In Section II, all 3 data sets have been split such that 90% is used as a training set and 10% as a test set. The data that goes into each set is chosen randomly using an algorithm that picks from a Gaussian distribution.

All input data points in each training set have been standardized to retain zero Gaussian mean and unit variance as follows:

$$x = \frac{x - \bar{x}}{\sigma}$$

x - data point, $\bar{x}$ - mean of data point, σ - standard deviation of data point

The *preprocessing* module in `Python` further provides a utility class *StandardScaler* that computes the mean and standard deviation on each training set so as to be able to later reapply the same transformation on their respective testing sets. Standardization is a very useful technique as it scales all features such that each feature has a relative proportional representation for a given hypothesis *h(x)*. If a feature has a variance that is orders of magnitude larger than others, it

[1]Python was used with `Tensorflow`, `scikit-learn`, `Keras` and other standard libraries.

[2]A more deeper analysis of the individual and combined data sets and their features are explored in the Appendix

might dominate in *h(x)* and make the estimator unable to learn from other features correctly as expected.

Furthermore, when using Ridge Regression the effect of λ is much bigger for standardized data components that have smaller energy compared to non-standardized data components. Moreover, gradient descent converges faster when using standardized input features[6].

D. Predictors and Loss function

In Section II, Linear Regression has been determined as an appropriate baseline predictor since the required output is real-valued. The Squared Error between predicted and test outputs has been chosen as the loss function, as it penalizes larger errors by making them larger and favors smaller errors by making them smaller. The loss function is hence defined as: $\ell(\hat{y}, y) = (\hat{y} - y)^2$ where $\hat{y}$ is the predicted output and y is the correct output from the test set. Thus the error measure that will be used to asses and compare performance between models will be the Mean Squared Error (MSE): $\frac{1}{n}\sum_{i=1}^{n}(\ell(\hat{y}, y) = (\hat{y} - y)^2)$ where n is the number data points in the test set.

II. BASELINE LINEAR REGRESSORS

A. Linear Regression

Linear Regression seeks to find a hypothesis such that the MSE is minimized. It does this by finding the optimal weight vector **Wlin** that minimizes the MSE. **Wlin** is calculated by solving the following expression[7]. The ordinary least squares linear regression class from the `scikit-learn` library was used to train and test for a optimal hypothesis, which was calculated using an SVD (Singular Value Decomposition) algorithm to find the Moore-Penrose Pseudoinverse X+, which is then multiplied by y to find wlin and hence the optimal hypothesis for this model[8].

The plot below shows the variation of predicted and actual test values when using linear regression on the combined data set.[3] The performance of the Linear Regression model is summarized Table 1.

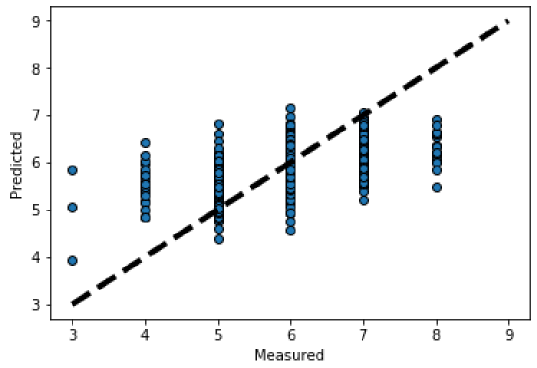


Figure 1: Predicted Vs Measured Test Data Points for Linear Regression

[3]The plots for RWD and WWD are in the Appendix.

B. 10 fold Cross-Validation and Regularization

To potentially build on the performance on this baseline predictor, other linear models which incorporate regularization such as Ridge Regression (L2)[9][10], LASSO (L1)[11][12], LASSO LARS (L1)[13][14] and Elastic Net (L1 + L2)[15][16] have been explored. 10-fold Cross Validation[17] has been used to determine the optimal regularization parameters that minimize the loss function and hence the expected test error for each respective model. Each model is then re-trained on the whole training set using their optimal regularization parameter and then their performance is re-tested and compared.

Figure 2 for CDS, portrays how the cross-validation score (accuracy) varies with different values of α for different models. As α tends to 0, LASSO, LASSO LARS, and Elastic Net score improves. As α tends to 0.1, Ridge Regression score improves.

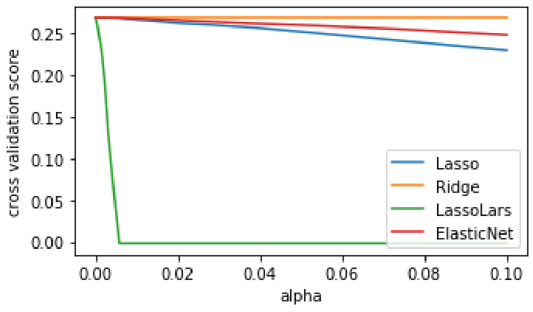


Figure 2: Cross-Validation Score Vs α Value

C. Performance Outcomes

Table 1 gives a summary of the performance of each model on each data set when using the optimal regularization parameters determined from 10-fold cross validation.

	DATA SET					
	Red Wine		White Wine		Combined	
MODEL	MSE	α	MSE	α	MSE	α
Linear	0.442	-	0.558	-	0.517	-
Ridge	0.442	0.1	0.558	0.1	0.517	0.1
LASSO	0.441	0.01	0.559	0.001	0.517	0.001
LASSO LARS	0.440	0.0001	0.561	0.0001	0.518	0.0001
Elastic Net	0.439	0.01	0.558	0.001	0.517	0.001

Table 1: Performance Summary of Linear Models

Table 1 shows that Linear Regression still has the best MSE for the combined and white wine data sets, hence regularization does not improve the accuracy of their baseline predictors. However, Elastic Net (L1-L2 split evenly) makes a slight improvement of 0.03 for the red wine dataset, and can achieve a 56% accuracy.

III. NEURAL NETWORKS

A. Theory - Artificial Neural Networks

In this section, more advanced methods such as Artificial Neural Networks (ANNs) have been explored, in order to determine if they can improve upon the performance of the linear predictors in section II.

It can be advantageous to use ANN's in relatively large data sets (greater than 1000 samples) due to their ability to adapt to complex non-linear dependencies, whilst still generalizing well to the test error. In this report, Feed-forward ANN's have been used.

Starting from the input layer, each neuron with activation function $\theta(x)$ in a given layer is connected to every neuron in the next layer up to the output layer[18]. Furthermore, there is no connection among neurons in the same layer, hence information is "fed forward" to subsequent layers. Each 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} .

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

The optimal accuracy for a given set of hyper-parameters can be determined by minimizing the loss function in weight space, which is implemented by using the back-propagation algorithm during training[19].

B. Implementation & Hyper-Parameter Tuning

In order to construct and configure an optimal ANN, various hyper-parameters had to be taken into account. Ideally, a coarse grid search [20] consisting of all applicable hyper-parameters in reasonable ranges would undergo cross-validation. This would determine the optimal values of the hyper-parameters for this data, since they are usually dependent on each other. However, due to time and restrictive computational resources, grid-searches were limited to 1-2 hyper-parameters at a time in order to get a reasonable estimation of ideal hyper-parameters.

The ANN was trained on the CDS with reduced features (9 features) as described in Section II. The input layer has 9 neurons corresponding to 9 input features. The number of hidden layers along with other hyper-parameters have been determined using *GridSearchCV* as described below. The output layer has one neuron with a linear activation function, since a real-valued output is required. Therefore, MSE can be used compare performance between the ANN and the linear models.

GridSearchCV takes in a dictionary of arrays of hyper-parameters. For each possible combination of the hyper-parameters the cross-validation score based on MSE of the ANN is determined. Once every combination is tested, the best combination of hyper-parameters that give the optimal cross validation score can be chosen. These 'optimal' hyper-parameters are then used to train the ANN on the whole training set. The MSE of this trained model is then calculated when trying to fit the test-data.

Pre-Determined Hyper-Parameters: The following hyper-parameters have been determined in advance and have been used when carrying out GridSearchCV on 1)The number of neurons, 2)The number of hidden layers and 3)The choice of activation function, unless stated otherwise.

- *Weight Initialization*: In order to ensure proper learning, weight vectors were initialized randomly from a normal distribution (Multivariate Gaussian)[21]. The *kernel_initializer = "normal"*[22] parameter normalizes the activations of the previous layer neurons at each batch by applying a transformation that ensures the mean activations are close to 0 and the activation standard deviations are close to 1. This ensures symmetry[23] and empirically improves the rate of convergence.
- *Optimizer*: The 'Adam' first order stochastic gradient descent method was used to calculate weights in the ANN, since is is computationally efficient and has good theoretical convergence properties[24]. A learning rate of 0.001 was used in conjunction with this optimizer, in order to determine how much to update the weight at the end of each batch.
- *Dropout*: A dropout of 0.5 was applied on the fully connected hidden layers, meaning that half of randomly selected neurons are ignored during training. Therefore, their contribution in the forward pass to the activation of neurons further downstream are temporally removed and corresponding weight updates are not applied to those neurons on the backward pass.
- *Regularization*: Following from the linear model performance, elastic net regularization (L1 + L2) was used on the training and testing of the final ANN to observe what effect it had on performance, however it seemed to have a detrimental effect on performance so ended up being omitted.
- *Batch Size & Number of Epochs*: The batch size aides in optimization by defining how many data points to show to the ANN, before updating the weights. For batch training all of the training samples pass through the learning algorithm simultaneously in one epoch before each weight update. Using the assumptions defined above and an ANN with 1 hidden layer(ReLu Activation) of 50 neurons, the best CV score of 0.525 (MSE = 0.475) was achieved using 150 batch size and 100 epochs.

1) Number of neurons: The number of neurons was not expected to exceed 100 neurons where the network is potentially too complex and expected to over-fit the training data. The configuration of the ANN when determining the number of neurons was 9-h-1 (9 neurons in the input layer, h neurons in the hidden layer and 1 neuron in the output layer), with h = $[10, 20, 40, 60, 80, 100]$, where the activation function in the hidden layer was ReLU defined as $\theta(x) = max(0, \sum_{j=1}^{J} w_{ij}x_{ij})$.

Neurons	10	20	40	60	80	100
CV MSE	0.62	0.61	0.55	0.52	0.53	0.53

Table 2: No. of Neurons Vs Cross-Validation MSE

From Table 2, it was clear that 60 neurons for one hidden layer had the lowest cross-validation MSE, hence 60 neurons were used in hidden layers when implementing further optimizations. For 80 neurons and above the CV MSE is about the same, however having more neurons may lead to over-fitting the training data, thus tending to give a worser generalization. Therefore, 60 neurons became an even more plausible choice.

2) Number of hidden layers: : Hidden layers can be seen as a distillation layers[25] that aim to filter for important patterns in data, whilst leaving out redundant information. Hence, multi-layered ANN's are capable of simultaneously expressing various non-linear surfaces.

Layers	1	2	3	4
CV MSE	0.550	0.541	0.549	0.553

Table 3: No. of Layers Vs Cross-Validation MSE

In Table 3, after 2 hidden layers the CV MSE increased as the number of hidden layers increased. This can be perceived as apparent over-fitting of the data. If there are too few hidden layers, there is possible high training and high generalization error due to under-fitting, along with high statistical bias. If there are too many hidden layers, there is possible low training error but high generalization error due to over-fitting and high variance[26]. Therefore, 2 hidden layers were chosen to take this Bias-Variance trade off into account[27].

3) Activation function of hidden layers: : The choice of activation function can affect convergence in the training process as well as accuracy.

Activation	ReLU	tanh	Sigmoid	Linear
CV MSE	0.54	0.56	0.66	0.54

Table 4: Activation Function Vs Cross-Validation MSE

The right activation function can greatly accelerate the convergence of 'Adam' in first order gradient descent and possibly improve the testing or validation accuracy. ReLU was chosen as it has the lowest CV MSE and is computationally inexpensive. It has disadvantages, as it drops negative values and can overload a neuron with large gradients, thus subsequent gradients flowing through the neuron will remain at zero for the rest of training. Hence, other activation functions were explored such as tanh $\theta(x)$, sigmoid and standard linear activation.

IV. RESULTS

The final ANN model had a configuration of 9-60-60-1, with ReLu activation function for all layers apart from the final layer, which had a linear activation function. Both hidden layers had a dropout of 0.5, and had undergone batch normalization. The solver was 'Adam' with a learning rate of 0.001. The ANN model was trained for 100 epochs with batch sizes of 150.

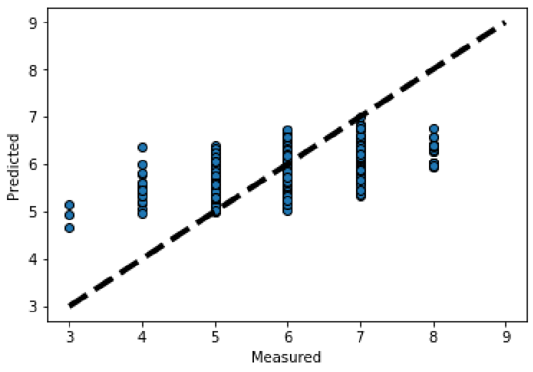


Figure 3: Predicted Vs Measured Test Data Points for ANN

The final ANN model achieved an MSE of 0.49 which is 51% accuracy on the test set, this is an improvement of 3% on top of the best linear model (Linear Regression) for the combined data set. When comparing Figure 3 with Figure 1, it can be seen that the ANN reduces the spread between predicted values, so the choice of output values are more definitive, even if the prediction is wrong.

V. CONCLUSION

As a result of the findings in this report, it was proposed to use the ANN over the Linear Models, due to the superior accuracy. However, if a linear model was chosen, then simple Linear Regression would most likely be the best choice based on the principle of Occam's Razor[28].

In Figures 1 & 3, for data points not in the 5-7 range there is significant error. This may be due to the fact that the majority of the data points in the CDS are centered around 5-7, and there is inherent sparsity as there is no 1,2,9 and 10 data-points. However, the imbalanced nature of the CDS with respect to the large difference in samples for the RWD and the WWD, along with the sparsity of some data labels could be addressed by simply collecting more samples for all labels for both wine types. This can lead to a data-set which is more balanced.

Future work which could not be encapsulated within the scope of this report, could include configuring, training and testing multi-classification Neural Networks and testing for Cross-Entropy Error. However, this work has been done before[29], but a more deeper Neural Network could be attempted. SVM Kernels that transform features into higher dimensions, could of uncovered unique classification boundaries in the data. As explained before, hyper-parameters would of ideally been explored in the most comprehensive manner, but this would needed extreme amounts of computational power to do in the given time frame.

A more interesting and perhaps a more fruitful method to pursue would be an ensemble method. For example, a consensus method that combines the optimal models from each respective algorithm class into an ensemble of classifiers can be attempted. An advantage of this approach is that labeled data is not required as clustering is used. Furthermore generalization performance is improved[29].

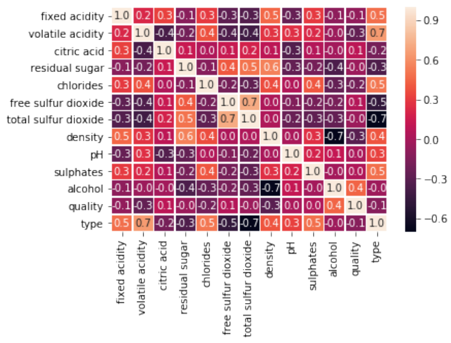
SUBMISSION PLEDGE

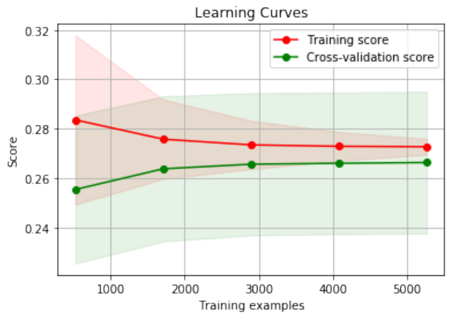
I, Rajan Patel, pledge that this assignment is completely my own work, and that I did not take, borrow or steal work from any other person, and that I did not allow any other person to use, have, borrow or steal portions of my work. I understand that if I violate this honesty pledge, I am subject to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

APPENDIX

A. *Combined Data Set Exploration*

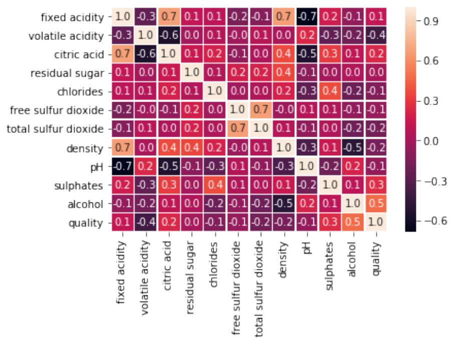
Correlation Matrix and Learning Curves

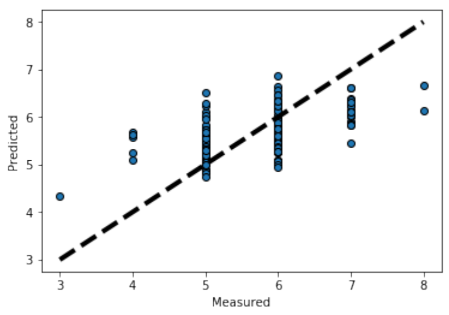


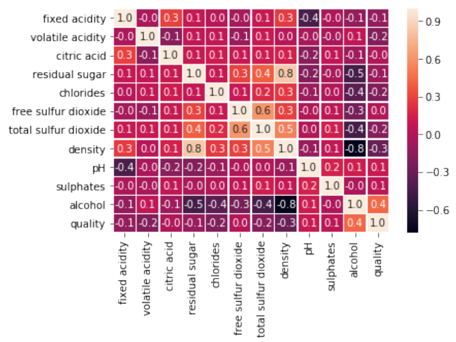


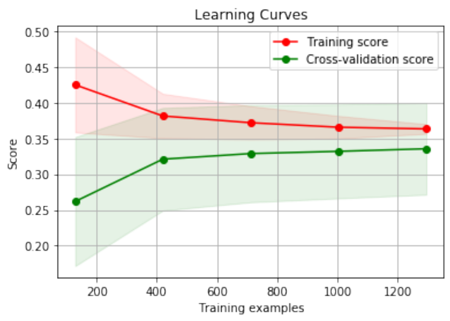
B. *Red Wine Data Set Exploration*

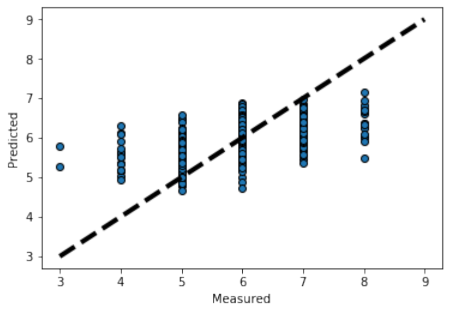
Correlation Matrix, Learning Curves and Regression Fit





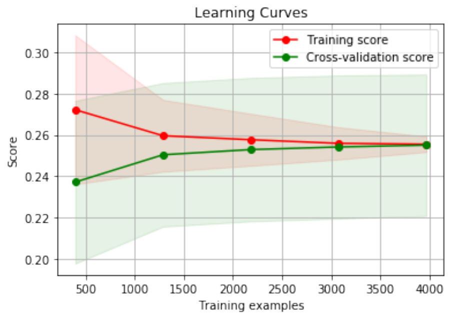






C. *White wine Data Set Exploration*

Correlation Matrix, Learning Curves and Regression Fit



References

[1] https://www.forbes.com/sites/tmullen/2018/02/15/state-of-the-wine-industry-2018-highlights-key-trends/26a04bc02de9
[2] http://www.ukva.org.uk/
[3] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.
[4] https://archive.ics.uci.edu/ml/datasets/wine+quality
[5] https://datastudentblog.wordpress.com/2014/01/30/the-influence-of-alcohol-content-and-volatile-acidity-on-wine-quality
[6] Lecture 5 Slide 35, Krystian Mikolajczyk 2018
[7] Lecture 2 Slide 12, Andras Gyoorgy 2018
[8] Lecture 2 Slide 13, Andras Gyporgy 2018
[9] Lecture 3 Slide 35, Andras Gyoorgy 2018
[10] Lecture 3 Slide 36, Andras Gyoorgy 2018
[11] Lecture 3 Slide 48, Andras Gyoorgy 2018
[12] Lecture 3 Slide 49, Andras Gyoorgy 2018
[13] Lecture 3 Slide 50, Andras Gyoorgy 2018
[14] Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
[15] Lecture 3 Slide 51, Andras Gyoorgy 2018
[16] Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
[17] Lecture 4 Slide 9, Krystian Mikolajczyk 2018
[18] Lecture 4 Slide 24, Krystian Mikolajczyk 2018
[19] Lecture 4 Slide 33, Krystian Mikolajczyk 2018
[20] Lecture 4 Slide 45, Krystian Mikolajczyk 2018
[21] Lecture 4 Slide 37, Krystian Mikolajczyk 2018
[22] https://keras.io/initializers/
[23] Lecture 4 Slide 37, Krystian Mikolajczyk 2018
[24] https://arxiv.org/abs/1412.6980v8
[25] https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207
[26] Geman, Bienenstock, and Doursat (1992)
[27] Boger and Guterman, 1997
[28] Learning From Data, YS Abu-Mostafa 2012
[29] Lecture 5 Slide 12, Krystian Mikolajczyk 2018