FYS-STK4155
Applied data analysis and machine learning
Project 2

Fred Marcus John Silverberg

12 October 2018

Abstract

The aim is to apply different machine learning methods on both regression and classification problems, with the purpose of identifying were a certain method excels and why. The model to be studied is the Ising model describing ferromagnetic phases and spin to spin interaction strengths with nearest neighbours.

I found that for the regression part, the linear regression methods impressed, all of which at some configuration yielded 1.0 R2 score while holding the MSE close to 0. Neural networks was not preforming well for this problem, as it had problems with an exploding gradient and only reached ≈ 0.90 R2 score with ≈ 4 in MSE.

The classification part was overwhelmingly dominated by the neural network, which achieved 100% accuracy score on the identification of ferromagnetic phases. Noted is that the neural network required sophisticated analysis and tuning before the accomplishment. A single layer with [neurons > 50] and [learning rate < 0.01] among other settings was found to work well for this data set. Applying this method on another set of data will certainly require another analysis of these parameters.

Contents

1 Introduction

The Ising model hold spin configurations that represent phases of ferromagnetism. As the temperature rise, energy increases and at a certain level, the spins are free to break from their ordered state. The objective for this project is to apply advance machine learning algorithms with the goal of accurately classifying a certain phase in two dimensions, as well as successfully predict the coupling constants in between the spins for one dimension.

With the expansion of computational power, the earlier theory of back-propagation rose from the shadows and with it the method of neural network became available for the simple man. The method can appear as a magical tool for successfully achieve the mentioned challenges. It has already yielded impressive results in fields of artificial intelligence, self driving cars and tailored ads. Neural networks are of particular interest since they can be applied in many other fields such as medicine, economics or social behaviour.

The project is build up in steps from simple linear regression to logistic regression and finally it takes on the neural networks. A deeper analysis regarding the calibration of the later is to be presented.

2 Theory

2.1 Ising model

In 1D the coupling constants J can be seen as an array of binary values (0 or 1). For this part we construct data such that the energy and states are known. Concentrating on the nearest neighbouring interactions, the situation can be written as:

$$E[\hat{S}] = -J \sum_{j=1}^{N} s_j s_{j+1} \tag{1}$$

Where $S_j = \pm 1$ is considered the spin variables. E is the energy and J denotes a particular coupling constant. This can be transformed into a linear regression problem:

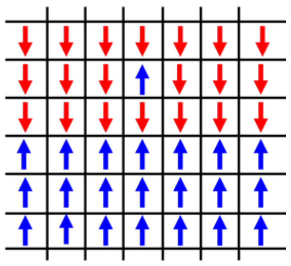
$$E = X^T J \tag{2}$$

With X being our matrix of energies and J a vector of coupling constants.

In 2D phase transitions are allowed, which occur at a certain critical temperature. The subject for this project will be to apply logistic regression and neural networks for determining a specific phase of a 40x40 lattice sheet of spins. The following configuration is to be observed with prepared data from Metha et al. [1]

$$E_{model}[S^i] = -\sum_{j=1}^{N} \sum_{k=1}^{N} J_{j,k} s_j^i s_k^i \tag{3}$$

Figure 1: Lattice of spins



Source: `https://bdhammel.github.io/2017/06/10/ising-model.html`

2.2 Linear regression

Linear regression can describe an output prediction $\hat{y}$ as a linear combination of a design matrix holding predictive variables, $\hat{X}$ and a set of coefficients $\hat{\beta}$. It is suitable for a target that is an interval varaible. Expressed in matrix form with error term:

$$\hat{y} = \hat{X}\hat{\beta} + \hat{\epsilon} \tag{4}$$

In this project the design matrix holds the spin configurations of the Ising model while the coefficients will represent the coupling constants ($\hat{\beta} = \hat{J}$). The regression methods for optimizing the coupling constant will be ordinary least square (OLS), Ridge and Lasso. The subject is to minimize respective cost function by taking the derivative with respect to $\hat{\beta}$ and set it to zero. The optimal solution is then obtained by solving for $\hat{\beta}$.

OLS

Cost function:

$$C(\hat{\beta}) = (\hat{y} - \hat{X}\hat{\beta})^T * (\hat{y} - \hat{X}\hat{\beta}) \tag{5}$$

Optimal solution:

$$\hat{\beta} = (\hat{X}^T\hat{X})^{-1} * (\hat{X}^T\hat{y}) \tag{6}$$

Ridge

Cost function:

$$C(\hat{\beta}, \lambda) = (\hat{y} - \hat{X}\hat{\beta})^T * (\hat{y} - \hat{X}\hat{\beta}) + \lambda\hat{\beta}^T\hat{\beta} \tag{7}$$

Optimal solution:

$$\hat{\beta} = ((\lambda)I + \hat{X}^T\hat{X})^{-1} * \hat{X}^T\hat{y} \tag{8}$$

Where λ is the penalty that limits the domain where $\hat{\beta}$ can express them self.

Lasso

Cost function:

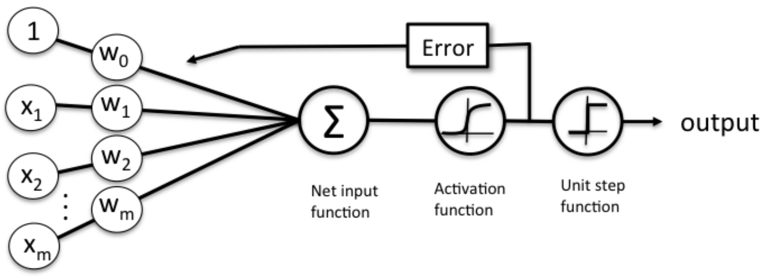
$$Cost(\hat{\beta}, \lambda) = \frac{1}{n} \parallel \hat{y} - \hat{X}\hat{\beta} \parallel_2^2 + (\lambda) \parallel \hat{\beta} \parallel_1 \tag{9}$$

Here λ is the penalty that shrinks the coefficients $\hat{\beta}$. Once again, the cost function is derivated and set to zero, where the optimal solution usually is obtained by an iterative solver.[2]

2.3 Logistic regression

Logistic regression is primary used for classification problems, where the target is a discrete variable. The method measure the relationship between the dependent variable and the independent variables by estimating probabilities. This is possible due to the logistic function, which transform the relationship into a number between (0,1). This number can in turn be transformed into a binary value by a threshold classifier, which determine if the value is high enough for a certain class. A figure is included below that illustrate the method.

Figure 2: Logistic regression model



Source: `https://rasbt.github.io/mlxtend/user_guide/classifier/LogisticRegression/`

By initializing random and normal distributed weights W_i the method multiplies them with the input variables X_i, which are then summed and activated by the logistic function, introducing non-linearity into the model. An error measurement is done and used for adjusting the weights. This process is repeated and finally the resulting probabilities goes through the threshold classifier for a final output. Imagine a forward and backward calculation encapsulated in an iteration process, then it can be further described as follows:

Cost function
The model is assigned a suitable cost function, which is to be minimized throughout the process. In this project the quadratic is for prediction and the cross entropy for classification.
Quadratic:

$$Cost(\hat{W}) = \frac{1}{m}\sum_{i=1}^{m}(yp - y)^2 \tag{10}$$

Cross Entropy:

$$Cost(\hat{W}) = \frac{-1}{m}\sum_{i=1}^{m}(y_i * ln(yp) + (1 - y_i) * ln(1 - yp)) \tag{11}$$

Where y is the target vector holding observed values, while yp is a function of $f(\hat{X}, \hat{W})$ and represents the predicted output vector, both with elements represented by i.

Forward phase
With a cost function defined, the first step is to calculate the net input value of the randomized weights assigned, where W_o can be thought of as the bias b.

$$Z = netinput = \sum_{i=1}^{m} X_i \bullet W_i + b \tag{12}$$

The net input values are then put into the activation function, obtaining probability values. There is many functions to choose from, in this project the standard will be the sigmoid function.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{13}$$

Back-propagation phase
An error calculation is made and used by the model to adjust the weights. This is usually done by using a gradient decent method. The chosen method here is the Stochastic Gradient Decent (SGD) with mini batches, which allows us to broadly search for the minima of our cost function. Instead of taking small but accurate steps towards the minima, the SGD rushes and stumbles across the cost function until it finds the treasure. This method decrease the computational time, which could otherwise be very heavy, as well as decrease the probability of obtaining a local minima instead of the global minima. For the first iteration, the gradient will to some accuracy, be in the direction of the steepest descent.[3]

With cross entropy chosen as cost function, the new weight value will be:

$$W' = W - \eta * \frac{\partial CW}{\partial W} \tag{14}$$

Where η is the regularization term, decreasing the probability of over-fitting. By using the chain rule, the partial derivative is written as:

$$\frac{\partial C(W)}{\partial W} = \frac{\partial C(W)}{\partial yp} * \frac{\partial C(yp)}{\partial Z} * \frac{\partial C(Z)}{\partial W} \tag{15}$$

Expanding the first term as the derivative of our cost function w.r.t prediction:

$$\frac{\partial C(W)}{\partial yp} = -\frac{y}{yp} + \frac{1-y}{1-yp} = \frac{yp-y}{yp(1-yp)} \tag{16}$$

Expanding the second term as the derivative of prediction, which is the same as the activation function of (Z) w.r.t (Z) :

$$\frac{\partial C(yp)}{\partial Z} = activationfunction(Z)' \tag{17}$$

For Sigmoid that is:

$$\frac{\partial C(yp)}{\partial Z} = yp(1-yp) \tag{18}$$

Expanding the third term as the derivative of net input (Z) w.r.t weights:

$$\frac{\partial C(Z)}{\partial W} = X \tag{19}$$

Calculating each partial derivative and simplify it results in:

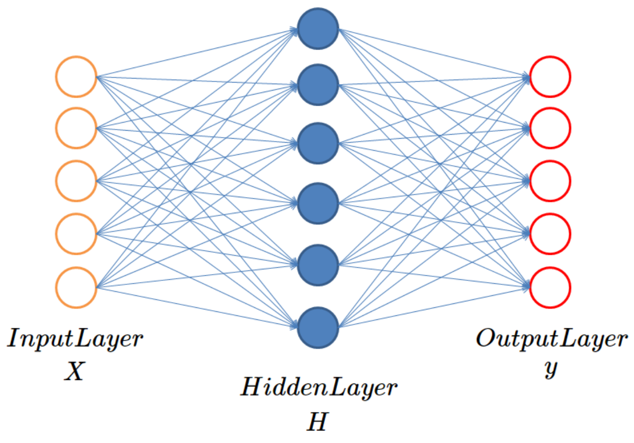
$$\frac{\partial C(W)}{\partial W} = (yp-y) * X \tag{20}$$

Adding the regularization term avoiding over-fitting: $W = W * (-\eta)$
The updated weights are then:

$$W' = W * (yp-y) * X \tag{21}$$

2.4 Neural network

Figure 3: Logistic regression model



Source: `https://chunml.github.io/ChunML.github.io/project/Creating-Text-Generator-Using-Recurrent-Neural-Network/`

A neural network is constructed as shown above. Here the output layer contains more than one output, if wanted. And one hidden layer is added, constructing a net of weights (blue lines) and biases. Additional hidden layers can be included. Each hidden layer can be assigned arbitrary many neurons (the blue circles). The idea is sprung from the logistic regression and extended into a more complex configuration of interconnected elements that processes the information dynamically. Further break down of the concept is as follows, with the encapsulated iteration process in mind:

A suitable cost function is assigned to the model.

Forward phase
Weights and biases are created randomly and normally distributed. The net input value (Z) is then calculated as:

$$Z = X \bullet W + b \tag{22}$$

Noted is that the calculations now take place as nestled matrices, where the matrix W represent all weights on both sides of a calculated layer, hence have the size (2,).

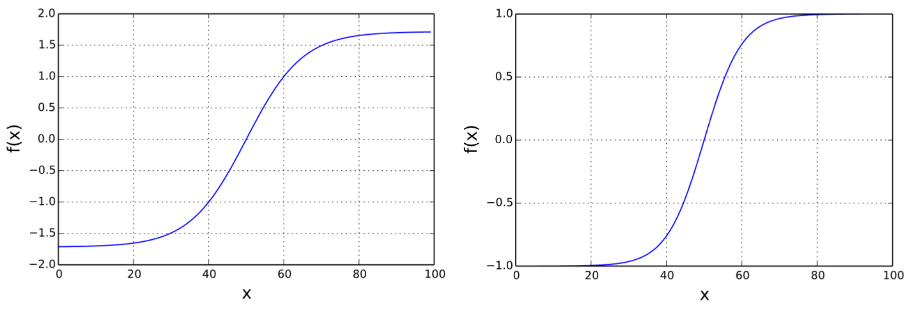
Clarity:
For figure (3): $W(0)_{ij}$ is the matrix of size (Inputs,Neurons) where i denote the input X_i and j denote the neuron H_j, hence holding all weights between the input layer and the hidden layer. While $W(1)_{jk}$ is the matrix of size (Neurons,Outputs) where j denote the neuron H_j and k the output O_k. Which corresponds to the weights between the hidden layer and the output layer. Similary for the bias b, which is a matrix (2,) that hold vectors of biases for each layer. $b(0)$ hold the biases for the hidden neurons and $b(1)$ the biases for the outputs

Z is then activated by chosen activation function for the interaction between the input layer and the hidden layer, and similarly by the hidden and output layer.

$$a(Z) \tag{23}$$

Figure 4: Activation functions

(a) Sigmoid. (b) tanh.



Back-propagation phase
Here the golden mathematics take place. First an error estimate is drawn δ_3, which is then passed back by the back-propagation algorithm for adjustments of the weights and biases. The gradients are derived with the same analogy as in the section of logistic regression with a stochastic gradient decent method. For cross entropy as choice of cost function and Sigmoid as activation function for the output layer, the back-propagation algorithm is implemented as below:[5]

$$\delta_3 = (yp - y) \tag{24}$$
$$\delta_2 = \delta_3 \bullet W_2^T * (a(Z_1))' \tag{25}$$
$$\frac{\partial C}{\partial W_2} = a(Z)_2^T \bullet \delta_3 \tag{26}$$
$$\frac{\partial C}{\partial b_2} = sum(\delta_3) \tag{27}$$
$$\frac{\partial C}{\partial W_1} = X^T \bullet \delta_2 \tag{28}$$
$$\frac{\partial C}{\partial b_1} = sum(\delta_2) \tag{29}$$

Layers are representative by (1,2,3) = (I,H_1,O) such that the recipe can be extended if additional hidden layers are added, as (1,2,3,4) = (I,H_1,H_2,O). Note that if the cost function or the activation function are changed, one would have to derive δ_3 from scratch.

A pseudo code of the Neural Network with SGD:

```
for i in range(epochs):
   * Construct mini batches
   for j in range(batches):
      * Draw mini-batch
      * Forward phase
         (Activates all neurons and predict outputs)
      * Back-prop phase
         (Estimate error)
         (Calculate gradients)
         (Update weights and biases with regularization)
Final prediction or classification
```

2.5 Calibration of Neural Network

There must be highlighted that the neural network are prone to some parameters that governs entirely how well the model will preform.[6]

Learning rate
A hyper-parameter that controls how much the weights and biases will adjust due to the gradient of the cost function. A small value could be described as taking small but accurate steps towards the minimum. For a large value, the path towards the minimum would be rough. To large and there is a probability of never reaching the precise minima, while to low may be stuck at a plateau. It also affects the computational time, a large learning rate would decrease the computational time and vice verse.

Regularization term
By applying a regularization term, the weights and biases (or coefficients for the linear case) is basically programmed to be small, with the assumption that a lower variance between them would represent a simpler model, hence it is a way of avoiding overfitting.

Epochs
Denotes how many times a training set is run through the neural network, both forward and backward. By increasing the number of epochs one would expect the model to obtain optimal weights and biases, only running it once would under-fit the model.

Batch size
Included in the SGD method, the training data is divided into smaller batches which run separately through the neural network. So instead of running the whole training set in one epoch, all batches runs once for one epoch. This allows us to run larger data sets. Optimal batch size is discussed but the size will effect the convergence of the gradient. Generally a small batch size in the range of 32-512 is to prefer.[7]

Layers and Neurons
Increasing the number of neurons or layers also increase the networks ability to learn detailed parts of the data, so for a complex situation more neurons and / or layers would be necessary. This can be adjusted along the way, by doing a grid-search for different numbers and compare the final score. The configuration also have impact on the proper choice of activation function.

Activation function
The choice of activation function will have impact on how the model preform as it transform the values into different domains. They may have steeper gradients and lower computational time, or be able to avoid vanishing gradients allowing deeper nets, or decrease the probability of exploding cost functions. For a network with multiple layers, one could use different functions for each layer.

2.6 Model assessment

[Below, y represents the target while yp represents the prediction.]

Mean square error:
MSE is the average of square errors, the difference between the observed values and the fitted values. Each difference is squared such that negative and positive values can be handled without interfering. A small value is preferred, since MSE = 0 is regarded as a perfect fit. Noted is that $MSE = Bias^2 + Variance + Noise$

$$MSE = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - yp_i)^2 \tag{30}$$

R2-Score:
Describe the total variance by the fitted values divided by the total variance of the observed values. Hence giving a picture of how well the fitted model correlates to the observed. The value spans between [-inf,1] where 1 represents perfect correlation.

$$R^2 = 1 - \frac{\sum_{i=0}^{n-1}(y_i - yp_i)^2}{\sum_{i=0}^{n-1}(y_i - E[yp])^2} \tag{31}$$

Accuracy Score:
This measure the number of correct classifications. Which is used for evaluating the network. If all elements are classified correctly the score will be 100 %.

$$A\text{score} = \frac{1}{n}\sum_{i=1}^{n} I * (y_i = yp_i) \tag{32}$$

Where I is the indicator function, 1 if $y_i = yp_i$ or 0 if $y_i \neq yp_i$.

Bias:
Bias is a measurement of the difference between the actual target value and the predicted value for a specific x input. It may give indication whether or not a model is over-fitted or under-fitted. A high bias value indicates the later.

$$Bias^2 = \frac{1}{n}\sum_{i=1}^{n}(E[yp] - y_i)^2 \tag{33}$$

Note that the expectation (E) values are for different sets $[x0, x1, x2...xn]$,$[y0, y1, y2...yn]$ out of the training set: $P(x, y)$. Where yp and y are functions of x.

Variance:
Variance measure the difference between the average predicted value and each single predicted value $yp_i = f(x_i)$ out of the set of x inputs. It describe the spread between the output values, a high value may indicate an over-fitted model and vice verse.

$$Variance = \frac{1}{n}\sum_{i=1}^{n} E[yp_i - E[yp])^2 \tag{34}$$

Resampling:
A tool that is used on limited data. It allows the analyser to draw more information out of the data available, hence evaluating the errors further and identify the variability without calculating the covariance. It is computationally heavy since the idea is to repeatedly train on one piece of data while testing on the left out piece. Bootstrap is here the chosen among many techniques. The concept is to randomly draw new datasets from the training data, with replacements. Then preform a new prediction and new statistics. This is done n times and is later averaged. The results may indicated whether or not the model used is to complex (high variance in the training set) or to simple (high bias in the training set).

3 Code implementation

The calculations and visualizations are to be preformed in python with usage of numpy and scikit-learn among other library's. The project will have the aim of object oriented coding, with classes handling most of the algorithms.

```
Main file:
  * project_2.py

Classes:
  * Linear_reg.py
    Handle input data and call linear regression as well as measurements.
  * Neural_Network_act.py
    Handle input data and preform heavy calculations.

Imported Files:
  * regressions.py
    Hold the code for all regression methods except the neural network.
  * logmethods.py
    Hold various pyhton methods from plotting to assesments.
```

4 Results

For the one dimensional case, the objective was to identify the coupling constant J with linear regression methods (OLS, Ridge, Lasso). The value was pre-set to ($J = -1$) The results below clearly demonstrate that the chosen methods are capable of achieving the purpose, since -1 appear as expected, along the central diagonals.

Figure 5: J matrices 1D

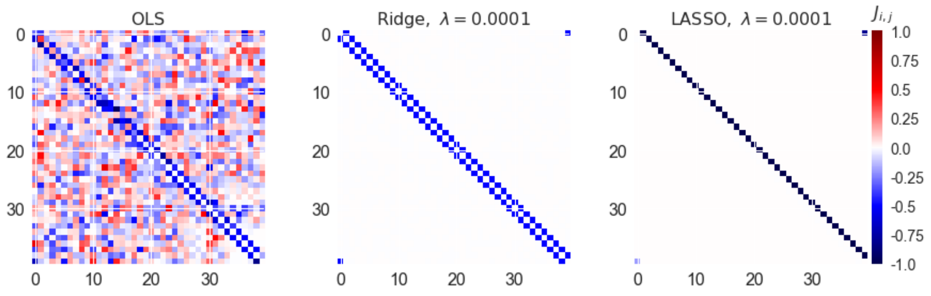


Figure (5): Illustration of the interaction matrices for the one dimensional Ising model, produced with test data on Sklearn:s OLS, Ridge and Lasso. Another pair of figures are found in the appendix.

Regarding the statistics, there was hints from figure (5) that they would be quite satisfying, which also was the case as shown below:

Table 1: Linear Statistics

Method	Variance	$Bias^2$	MSE	R2 score
OLS	0	$9.44471e-28$	$9.44471e-28$	1
Ridge	0	$5.60239e-06$	$5.60239e-06$	1
Lasso	0	$4.56662e-07$	$4.56662e-07$	1

Table 2: Resampled Linear Statistics

Method	Variance	$Bias^2$	MSE	R2 score
OLS	$1.1812e-27$	$2.47832e-29$	$1.20496e-27$	1
Ridge	$5.01141e-06$	$1.312e-05$	$1.81314e-05$	1
Lasso	$6.83042e-09$	$4.73088e-07$	$4.79918e-07$	1

Table (1,2): The statistics above were obtained with L1(Lasso penalty) = 0.0001, L2(Ridge penalty) = 1 and 50 bootstraps for the resampled statistics. All on test data.

The statistics for neural networks came as follows. Noted shall that the resampled statistics had the same pattern for an increasing number of epochs. For optimal weights and biases see appendix for link, since they are many.

Table 3: Statistics Neural Network

Configuration	Variance	$Bias^2$	MSE	R2 score
$[30, 20]$	0	3.79831	3.79831	0.909063
$[40, 20]$	0	4.42479	4.42479	0.894064
$[40, 10]$	0	3.49727	3.49727	0.91627

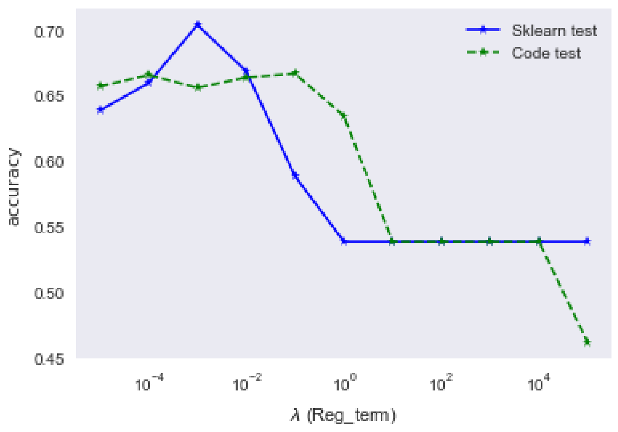
Table 4: Resampled Statistics Neural Network

Configuration	Variance	$Bias^2$	MSE	R2 score
$[30, 20]$	0	4.3754	4.3754	0.895247
$[40, 20]$	0	4.24964	4.24964	0.898258
$[40, 10]$	0	3.5556	3.5556	0.914874

Table (3,4): Settings: Learning rate = 0.0002, Reg term = 0.01, Batch size = 32, Epochs = 10, Bootstraps = 50. Sigmoid was used as activation function for the first layer, while identity function was used for the rest. All on test data.

The method of logistic regressions ability to classify a phase in the two dimensional Ising model is shown below, it is a bit better than average Joe, for penalties below 10. In this part, the penalty was the most sensitive parameter apart from the learning rate. The highest scores was obtained with the configurations below.

Figure 6: Accuracy score logistic regression



Figure(6): Illustration of accuracy score on test data. Obtained by batch size 32, learning rate for code = 0.001, and for Sklearn learning rate was set to 'optimal'.

Now, for the most interesting part of the project, the impressive accuracy score is here presented for a range of neural network configurations for the most sensible parameters. A fine calibrated network is able to classify a phase by a whopping 100%.

Figure 7: Grid search for Neural Network

(a) Learning rate vs Neurons (b) Learning rate vs Batch Size

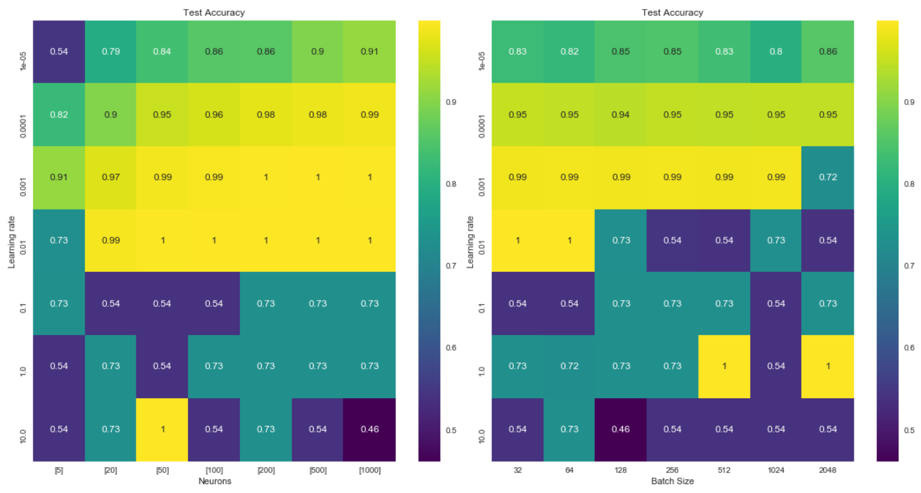


Figure (7): Settings: Reg term = 0.01, Epochs = 5, Batch size = 64, Learning rate = 0.001 with only sigmoid as activation function. Noted is that for the yellow outliers, the results are due to 'nan' computations and should be discarded.

5 Discussion

The identification of the coupling constants is obviously a simple task for the linear regression methods. OLS and Ridge solves the task in similar symmetry. The coupling constants are then expressed as pairs along the lower and upper three-diagonals with half the energy each, see figure(5). OLS gives a more scattered interaction matrix, but the pattern can still be identified, while Ridge isolates the constants in an excellent manner with close to zero off-diagonal elements. In a clinical way, the Lasso method solve for the coupling constants in the main diagonal, the accuracy increases with shrinking penalty.

The linear statistics (table 1,2) give the impression that the model is easy to fit, since there is no variance, close to zero MSE and perfect R2 scores. The resampling reveal that some variance:s are present, but not to an extent that the results should be reconsidered, as the MSE:s are still close to zero.

Applying neural networks to the one dimension model for obtaining the optimal weights and biases was an intensive fight with the computer. The gradient had a tendency to explode, especially for deeper networks. Normalizing would get me a few epochs further, but the cost was then paid in decreased score as the adjustments of weights and biases decreased in magnitude. I identified the learning rate and the batch size as parameters that were the most important to optimize for avoiding the problem and at the same time reach fine results, see table (3). Resampling of the data with neural networks gave some contradicting indications. The squared bias rose, and by that the MSE, for one type of neuron set-up, while decreasing for another, see table (4).

As for the classification part, the difference in accuracy score from the plain logistic regression and the neural network is baffling at a first glance, compare scores on figures (6,7). The neural networks gave impressive results for fine tuned configurations. By grid searching the different settings and comparing the corresponding accuracy scores, optimal values for the overall configuration could be obtained. The process itself, yielded the insights of how this 'magical' method still relies on parameters that we need to tune.

Apart from the number one sensitive parameter, the learning rate, the dependence of layers and neurons should be highlighted. For this particular problem, a single layer is enough for reaching close to perfect score. An increase of neurons are then of importance for gaining the remaining accuracy.

6 Conclusion

For the regression part of the Ising model, the methods of linear regression achieved the best results for prediction, were OLS achieved approximately zero MSE in combination of 1.0 in R2 score. The penalty dependent methods Ridge and Lasso, were best suited for the identification of the coupling constants along the diagonal of the interaction matrix [J]. A neural network came short here, as it did not preform as well with 0.90 in R2 score and with a computational effort that increases with complexity compared to the steady linear regressions.

The logistic regression proved itself to give better than random results for the classification of phases, 65% accuracy score, but nothing more spectacular. However, it is crystal clear that a neural network is suited for solving a classification problem. By building on the logistic idea and implementing artificial neurons that dance in an interactive process, governed by the beauty of mathematics, one can achieve 100% score. The most important drawback of this method is the sensitivity to configuration. An analysis of the parameters should be considered obligatory when approaching a new situation with this method. The need for computational power and time is another subject to consider.

For a more professional approach, the SGD could in the future be modified into a solver that continues until a defined minima of the cost function is found. An implementation of a dynamic learning rate or an other function that handles the exploding gradients, would also increase the credibility. There is also humongous many combinations of settings that can be further explored with other data sets and multi classification problems.

7 Appendix

Interaction matrices

Figure 8: Penalty L2,L1 = 10000

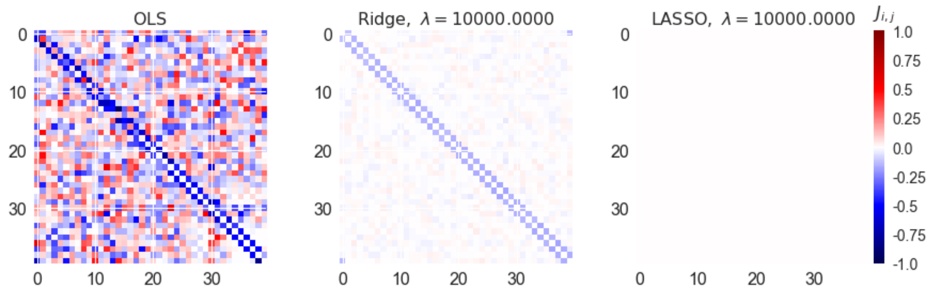


Figure 9: Penalty L2,L1 = 1

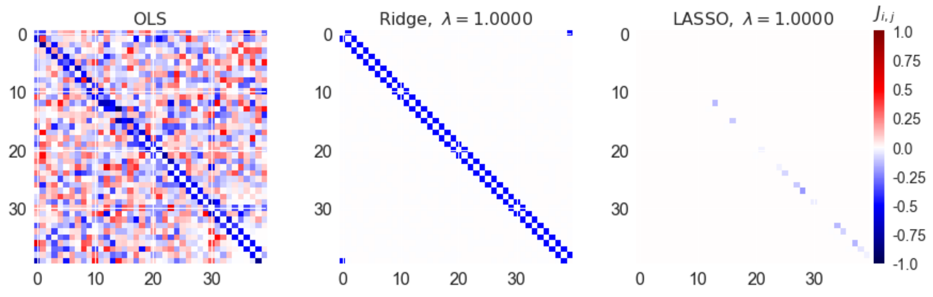


Figure (8,9): The figures highlights the strength at critical penalty values for the regularization methods Ridge and Lasso.

Optimal weights and biases

Biases [[40,],[20,],[1,]] array, as numpy file at:
`https://github.com/silverberg89/FYS-STK4155/blob/master/Project_2data/biases_optimal.npy`

Weights [[1600,40],[40,20],[20,1]] array as numpy file at:
`https://github.com/silverberg89/FYS-STK4155/blob/master/Project_2data/weights_optimal.npy`

Access to all material can be found at:
`https://github.com/silverberg89/FYS-STK4155/blob/master/Project_2data/`

References

[1] Pankaj Mehta. 2018. A high bias low-variance introduction to Machine Learning for physicists. [ONLINE] Available at: https://physics.bu.edu/ pankajm/MLnotebooks.html. [Accessed 8 November 2018]

[2] Morten Hjorth-Jensen. 2018. Computational Physics Lectures: Logistic Regression. [ONLINE] Available at: https://compphysics.github.io/MachineLearning/doc/pub/LogReg/html/LogReg.html. [Accessed 08 November 2018].

[3] Morten Hjorth-Jensen. 2018. Computational Physics Lectures: Logistic Regression. [ONLINE] Available at: https://compphysics.github.io/MachineLearning/doc/pub/LogReg/html/LogReg.html. [Accessed 08 November 2018].

[4] Morten Hjorth-Jensen. 2018. Data Analysis and Machine Learning: Neural networks. [ONLINE] Available at: https://compphysics.github.io/MachineLearning/doc/pub/NeuralNet/html/NeuralNet.html [Accessed 08 November 2018].

[5] Michael Nielsen. 2018. Neural Networks and Deep Learning. [ONLINE] Available at: http://neuralnetworksanddeeplearning.com/chap2.html. [Accessed 8 November 2018].

[6] Michael Nielsen. 2018. Neural Networks and Deep Learning. [ONLINE] Available at: http://neuralnetworksanddeeplearning.com/chap3.html. [Accessed 8 November 2018].

[7] Nitish Shirish Keskar. 2018. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. [ONLINE] Available at: https://arxiv.org/abs/1609.04836v2. [Accessed 8 November 2018].