Data Analysis and Machine Learning
Project 2

Andrei Kukharenka, Anton Fofanov and Anna Gribkovskaya
FYS-STK 4155

Abstract

This project is devoted to study of Ising model using different statistical methods. We begin with linear regression then move to classification of spin configurations using either logistics regression and neural network. LASSO regression provide the best results among the regression methods with $R^2 = 1$ and $MSE = 1 \times 10^{-6}$ for $\lambda = 1 \times 10^{-3}$. The best λ-values for LASSO have been found to be in the interval $[1 \times 10^{-1}, 1 \times 10^{-3}]$
For the logistics regression used for 2D Ising model we have obtained accuracy in range $[43\%, 68\%]$, that makes this method not a good choice for this problem.
The neural network applied to the 1D Ising model have been found able to simulate the linear regression results. For the 2D case neural network provide much better accuracy then logistics regression.

Contents

1 Introduction

Data science is one of the most rapidly developing parts of information technologies nowadays. The increase of computer power allow us to analyze huge amounts of data and this require some specific methods and techniques to be studied. Some of them have been already under consideration in the previous project, for example simple regression methods - linear, Ridge and Lasso. In this project we aim to tackle a classification problem using logistics regression. After that our goal is to move towards the neural network. The problem we are going to use as a test bed is Ising model.

Structure of the report. The first part is a theoretical description of the 1D and 2D Ising model. Second part is a brief description of the methods. After this we move toe the results and discussion part. The last part is conclusion where we present a brief summary of what have been dona and also discuss some possibilities for further research.

2 Problem description

This project is mostly based on the work of Metha et al. [1] and that's why we are using the problem formulation provided in this article. However, Ising

model is a well known model in physics and one may find many studies devoted to the model. For example, it's one of the natural choices to study Monte Carlo simulations, as it have been done here [3].
Ising model is a model that allows us to to compute the energy of a system of spins. Each spin can take only two values ± 1. The interaction is allowed only for closest neighbors. Generally speaking the Ising model provides us a simple approach to model the phase transitions of a ferromagnet. Phase transitions are transitions between ordered or disordered states. As soon as ordered state is more preferable for lower temperatures and disordered state is more preferred for higher temperature there shroud be some critical temperature when the switch is happening. In the project we will study 1D and 2D Ising models and the periodical boundary conditions are used. For the 1D Ising model there is no phase transition, while for the 2D the phase transition is happening at the critical temperature $T_c/J = 2/\log(1+\sqrt{2}) \approx 2.26$.

2.1 1D Ising model

The Hamiltonian for the classical 1D Ising model is given by

$$H = -J \sum_{i}^{N} S_i S_{i+1}, \qquad S_i \in \{\pm 1\}, \tag{1}$$

where N is number of particles in the system, and S_i is a spin pointing up or down.

2.2 2D Ising model

The Hamiltonian for the classical 2D Ising model is given by

$$H = -J \sum_{\langle ij \rangle}^{N} S_i S_j, \qquad S_j \in \{\pm 1\}, \tag{2}$$

where the lattice site indices i,j run over all nearest neighbors of a 2D square lattice, and J is some arbitrary interaction energy scale. Onsager proved that this model undergoes a thermal phase transition in the thermodynamic limit from an ordered ferromagnet with all spins aligned to a disordered phase.

2.3 Ising model for statistical methods

In order to apply the regression methods to the Ising model we need to rewrite the equations in terms of so-called coupling coefficient J_{ij} as:

$$H = -\sum_{\langle ij \rangle}^{N} J_{ij} S_i S_j. \tag{3}$$

Now we can apply regression to determine the J_{ij}. In order to use classification methods, for example logistics regression, we use already prepared 2D spin configurations that have been prepared by Metha et al. in [1].

3 Methods

Here we present a brief overview for the methods used on this project. Linear regression methods have been already presented in Project 1 [2], so we are now focusing on logistics regression and also will provide a theoretical description for neural networks.

3.1 Neural networks

Neural networks are very popular in the field of machine learning. The name "neural" refer to the fact that such networks are supposed to mimic a biological system of communicating neurons. Neural network is a network of layers each of them containing an arbitrary number of neurons. The connection in this case is represented by a weight.
The artificially build neural network should be able to behave similarly to a real neural network in human brains. By the analogy how the neurons connected in the human brain the mathematical model was established which can be used for both classification and regression problem. Starting from the simplest possible example the neural network can be represented as the following(see figure 1). The neural network has input and output layers, alongside with some amount

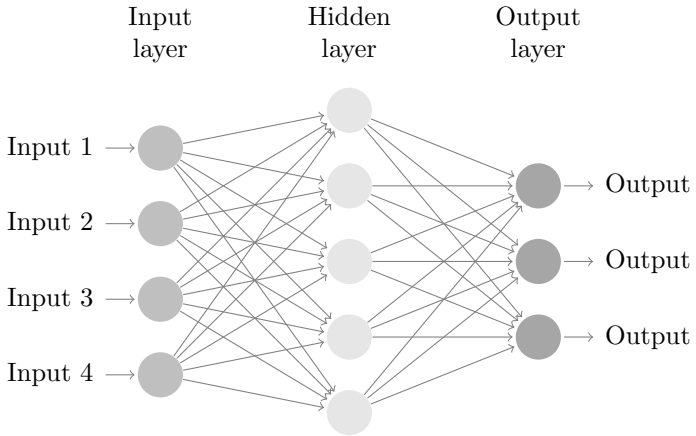


Figure 1: Neural network scheme.

of the hidden layers located between input and output. The way neural network operates is activations in one layer determine the activation of the next layer. It can be assigned a weight to each one of the connections between our neuron and the neuron from the following layer. Then the weighed sum of all activations

from the input layer is calculated. To be able to handle numerically the values of the activation sum(these values in general can have any values depending on the size of neural network), one needs to have these values to be in interval [0, 1]. To do this an activation function is used. A common function to use is the sigmoid function. The different activation functions we tested in this project will be described below in a greater details.
So the activation function is a basically a way to determine how positive this particular weighed sum is. To find a threshold for activation of one neuron one needs to add some number to the weighed sum. To be more precise, to subtract it from the weighed sum. This number is called the bias and it determines how large the weighted sum needs to be to activate a neuron. Described procedure was for just a one neuron from the hidden layer, where all the neurons from the input layer were connected with. This trail of thoughts can be repeated for the each neuron in the hidden layer, so each neuron has an individual weight and one dedicated bias. We need to find all the right weights and biases. Mathematically it can be expressed via linear algebra language. Organizing all the inputs into a vector A, such so for input layer($l = 0$):

$$A^{(0)} = \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_3^{(0)} \\ \vdots \\ x_m^{(0)} \end{bmatrix} \tag{4}$$

We can construct a weight matrix ($W^{(l)}$)associated with each layer, where each row represents a connection between one layer and a particular neuron in the next layer. By denoting via vector B a row-vector, containing all the biases, the activation of the next layer can be expressed via the following expression:

$$A^{(l+1)} = \sigma(W^{(l)} A^{(l)} + B) \tag{5}$$

For the input layer one applies the latter equation to the input vector and then its activation function. The second layer receives the output of the first layer and the whole scheme is repeated up to output layer.
The learning process of the neuron network is just an adjustment of the weights and biases. So when the neural network gets an unknown data as an input, the output will be correct. We define a cost function which might be seen as a representation of the correlation between input and output. Squared difference between desired result and the output of the neural network can be considered as a measure here. So the cost function is a multidimensional function of all the weights and biases which should provide us a score how good our neural network performs. It is naturally to employ some kind of multi-dimensional minimization algorithms to find a minimum of the cost function. The most simple approach is to use the steepest descent algorithm [1].

3.1.1 Activation functions

The sigmoid or so-called logistic function has been widely used in the hidden layers of the neural network:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{6}$$

As an alternative one might use the hyperbolic tangent as the activation function. Another example is the rectified linear unit (ReLU), which is motivated by a biological analogy(neurons would be rather activated or not):

$$\text{ReLU}(x) = \max(0, x)$$

The rectified linear unit usually performs the best [1]. In our project we use all mentioned activation functions.

3.1.2 Back propagation

Neural network is learning by updating the unknown quantities w_{ij} called weights and biases. The algorithm that allows us to update these quantities is called back propagation algorithm. For details please refer to [5]

3.2 Logistic Regression

Contrary to the regression methods which is focused on learning the coefficients of a polynomial fit in order to be able to predict the response for some unseen data the classification method is focused on outcomes taking the form of discrete variables. More specifically, the binary logistic regression used in this project is used for problems with two possible outcomes.
The probability for the event in this case is given by the sigmod function given in 6.
We assume now that we have two classes

$$\begin{aligned} p(y_i = 1|x_i, \hat{\beta}) &= \frac{\exp(\beta_0 + \beta_1 x_i)}{1 + \exp(\beta_0 + \beta_1 x_i)}, \\ p(y_i = 0|x_i, \hat{\beta}) &= 1 - p(y_i = 1|x_i, \hat{\beta}), \end{aligned} \tag{7}$$

The solution is found by maximizing the likelihood function

$$P(\mathcal{D}|\hat{\beta}) = \prod_{i=1}^{n} \left[p(y_i = 1|x_i, \hat{\beta})\right]^{y_i} \left[1 - p(y_i = 1|x_i, \hat{\beta}))\right]^{1-y_i}$$

which is used to obtain the cost function:

$$\mathcal{C}(\hat{\beta}) = \sum_{i=1}^{n} \left(y_i \log p(y_i = 1|x_i, \hat{\beta}) + (1 - y_i) \log \left[1 - p(y_i = 1|x_i, \hat{\beta}))\right]\right). \tag{8}$$

The cost function can be rewritten as a cross entropy:

$$\mathcal{C}(\hat{\beta}) = -\sum_{i=1}^{n} \left(y_i(\beta_0 + \beta_1 x_i) - \log\left(1 + \exp\left(\beta_0 + \beta_1 x_i\right)\right)\right). \tag{9}$$

The minus sing here is needed to find global minimizer. Now this is a convex function of some parameters and we can minimize the function:

$$\frac{\partial \mathcal{C}(\hat{\beta})}{\partial \beta_0} = -\sum_{i=1}^{n} \left(y_i - \frac{\exp\left(\beta_0 + \beta_1 x_i\right)}{1 + \exp\left(\beta_0 + \beta_1 x_i\right)}\right), \tag{10}$$

$$\frac{\partial \mathcal{C}(\hat{\beta})}{\partial \beta_1} = -\sum_{i=1}^{n} \left(y_i x_i - x_i \frac{\exp\left(\beta_0 + \beta_1 x_i\right)}{1 + \exp\left(\beta_0 + \beta_1 x_i\right)}\right). \tag{11}$$

In a more compact way this can be presented as:

$$\frac{\partial \mathcal{C}(\hat{\beta})}{\partial \hat{\beta}} = -\hat{X}^T \left(\hat{y} - \hat{p}\right) \tag{12}$$

To solve this we need to apply an iterative method, for example a gradient descent or stochastic gradient descent(GD). These methods are described in details here [4].
An important thing here is a learning rate. Any gradient method is based on initial guess which is then updated using a gradient of the function under consideration and some step. For example for some $\mathbf{X}$:

$$\mathbf{X}^{new} = \mathbf{X}^{old} - \eta \nabla \mathbf{X}^{old} \tag{13}$$

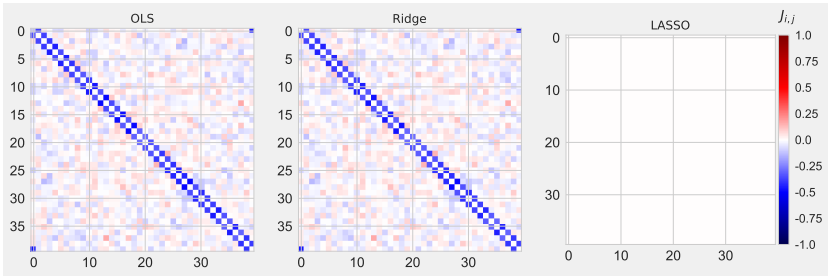
This step η is called learning rate. Gradient descent is extremely sensitive to the choice of learning rates.

4 Results and discussion

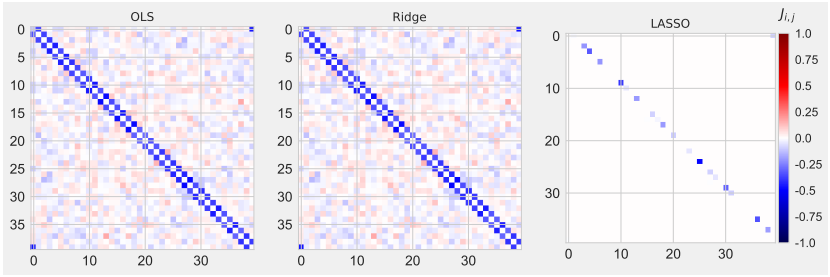
Here we present result starting with the 1D Ising model and linear regressions. Then we move to the neural network applied to the same system. After this we present results for 2D Ising model.

4.1 Linear Regression

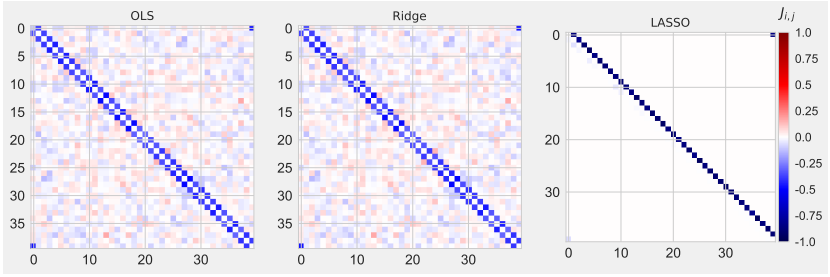
Figure 2 presents the heat maps for coupling parameters for different lambdas. Values along the diagonal represents particle under consideration and nearest neighbors are represented by left and right values. In our case only nearest neighbors have significant influence on each other. As we can see from the figures LASSO regression is the best fit in the case. OLS and Ridge does not seem to be effected by choice of λ and for LASSO regression the best λ-values are in the interval $[1 \times 10^{-1}, 1 \times 10^{-3}]$.



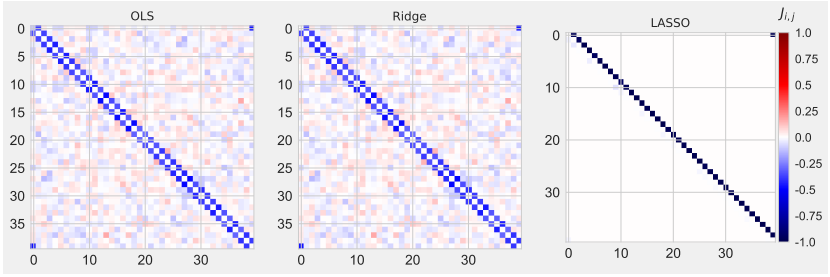
(a) $\lambda = 1.0 \times 10^{1}$



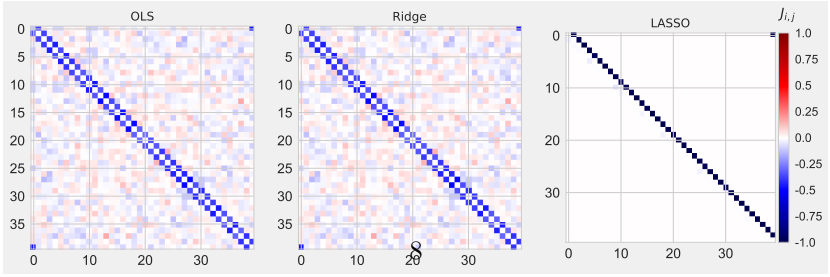
(b) $\lambda = 1$



(c) $\lambda = 1 \times 10^{-1}$



(d) $\lambda = 1 \times 10^{-2}$



(e) $\lambda = 1 \times 10^{-3}$

Figure 2: Heat map for coupling coefficient J_{ij} for different value of parameter λ and various regression methods.

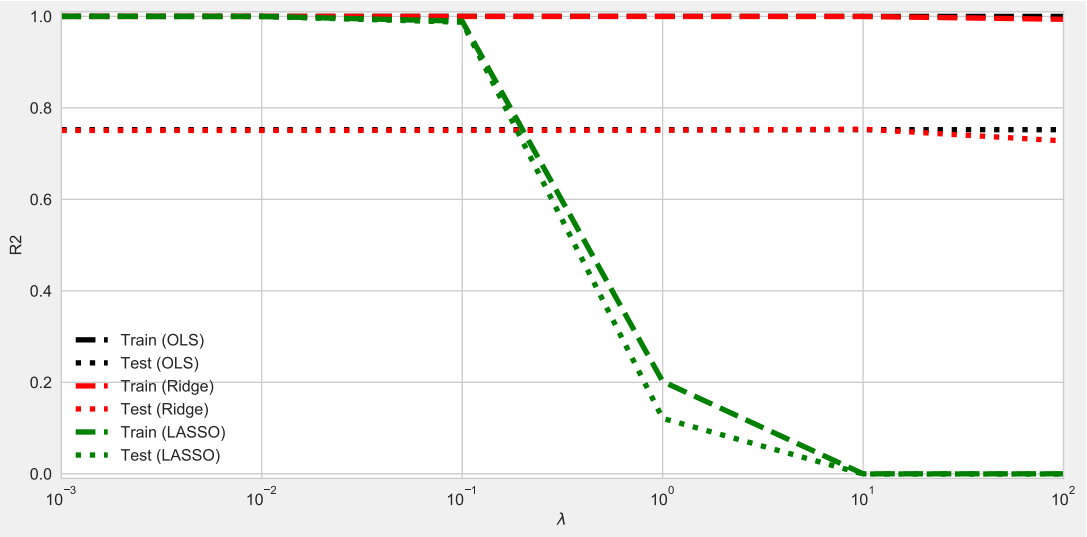


Figure 3: R2

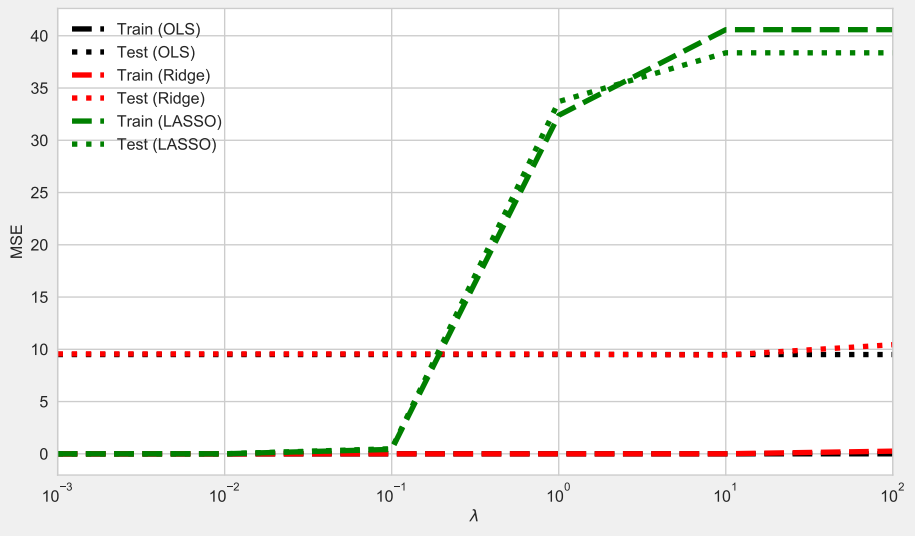


Figure 4: MSE

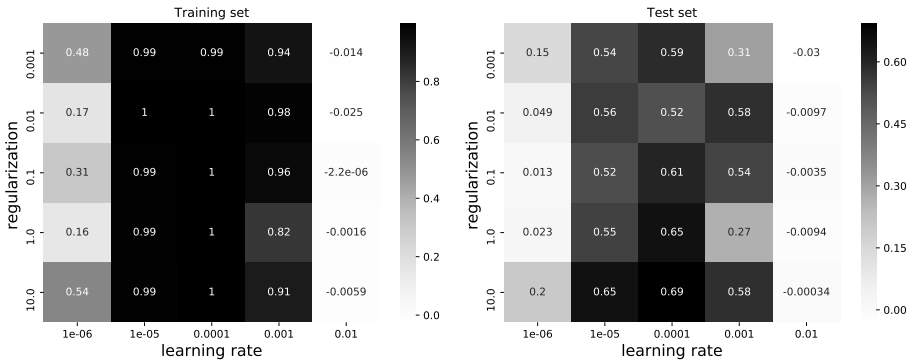


Figure 5: The R2 heatmap for N=4 neurons.

4.2 Neural network

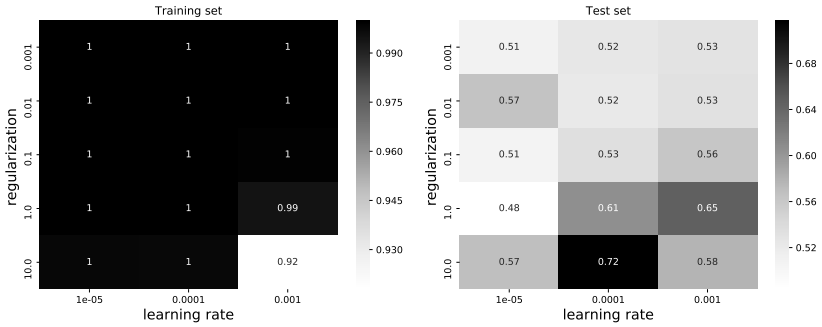
Figure 5 one can see a heatmap for just 4 neurons. As one can see it provide a good result for training data set, except for some values of learning rate. The optimal learning rate is in range $[1 \times 10^{-5}, 1 \times 10^{-3}]$.
Figure 6 present results in the optimal range for different numbers of neurons. As one can see the performance for training set is better then for the test set. This might be a sing of overfitting, however we cannot improve it by changing the values of λ or running the program for more neurons.

4.3 Logistics Regression

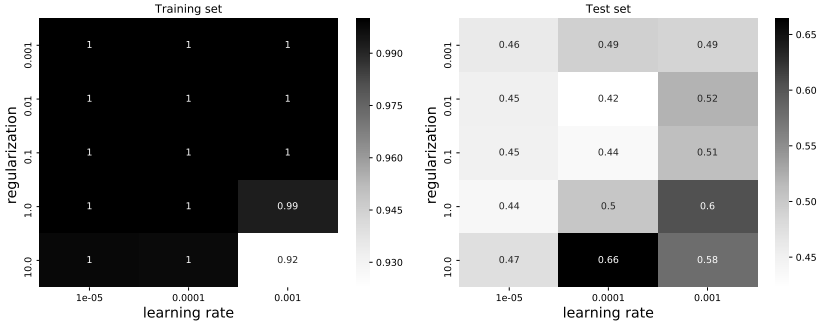
On Fig. 7 one can see the results for logistics regression. The method provide quite poor accuracy, it depends on the applied optimization method but mostly vary in range $[43\%, 68\%]$.

5 Conclusion

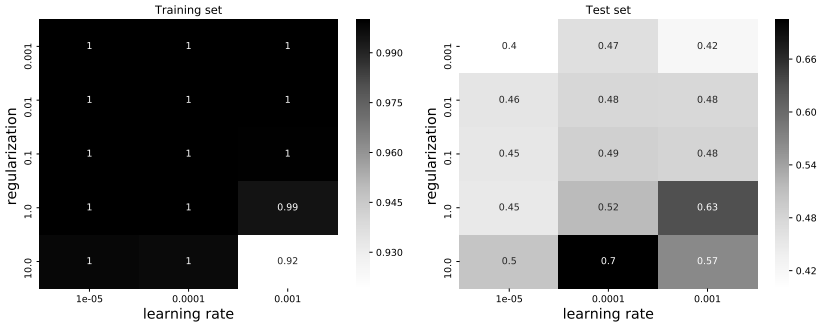
In this project we have studied various statistical methods for Ising model. We have applied OLS, Ridge and LASSO regression to the model and find out that all model may be used to represent the Ising model as they return the largest values along the diagonal. However, the LASSO model provide the best fit and also agrees with results obtained by Metha et al. in [1]. After this we try test the neural network on the same model and have found that for some values of regularization parameters it provides a good result on training data set.
After this we move to logistics regression on 2D Ising model and here we were not able to obtain a reliably accurate results. Accuracy depends on the applied method and parameters but mostly vary in range $[43\%, 68\%]$.



(a) N=64

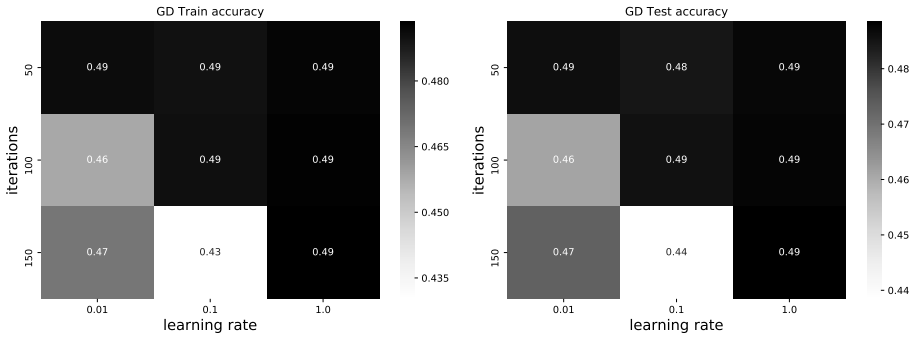


(b) N=128

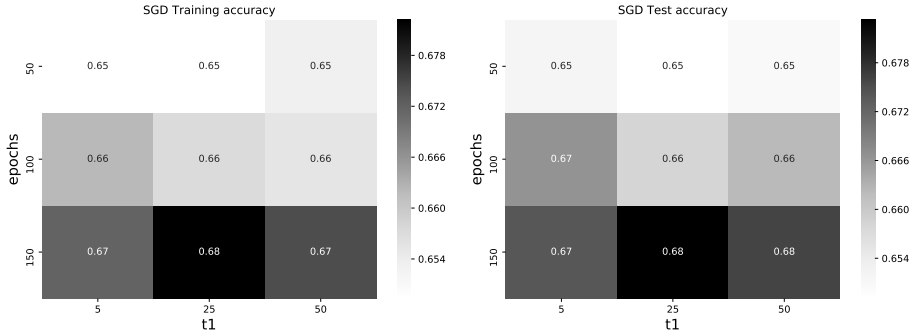


(c) N=256

Figure 6: The R2 heatmap for different number of hidden neurons.



(a) Gradient Decent



(b) Stochastic Gradient Descent

Figure 7: Logistics regression for various parameters

6 Further work

As we have found that performance of the neural network depend a lot on the various parameters it would be natural to continue examine their influence as we were not able to do properly in this project. Also we way try to increase the speed of the computations by using parallelization technique for the neural network.

References

[1] Pankaj Metha et al. *A high-bias, low-variance introduction to Machine Learning for physicists* ArXiv:1803.08823.

[2] Andrei Kukharenka et al. *Project 1. FYS-STK4155* https://github.com/andrei-fys/fys-stk4155

[3] Andrei Kukharenka, Anna Gribkovskaya. *Project 4. FYS4150* https://github.com/andrei-fys/fys4150/tree/master/Project˙4

[4] Morten Hjorth-Jensen *Data Analysis and Machine Learning Lectures: Optimization and Gradient Methods* https://compphysics.github.io/MachineLearning/doc/pub/Splines/html/Splines-bs.html

[5] Morten Hjorth-Jensen *Data Analysis and Machine Learning: Neural networks, from the simple perceptron to deep learning and convolutional networks* https://compphysics.github.io/MachineLearning/doc/pub/NeuralNet/html/NeuralNet.html