UiO : Department of Physics
University of Oslo

**FYS-STK4155: Applied Data Analysis and Machine Learning**

# Project 2

## Classifying Phases of the 2D Ising Model with Logistic Regression and Deep Neural Networks

Tobias Netskar Kristian Wold Nicolai Haug

November 8, 2019

### Abstract

In this project, we have applied several statistical learning methods to data from the Ising model for atomic spins. For the one-dimensional Ising model, a neural network and linear regression methods were used to determine the coupling constant for the Ising model Hamiltonian. The two-dimensional Ising model exhibits a phase transition from an ordered phase to a disordered phase at a critical temperature. The goal was to build a model with both logistic regression and neural networks, which for a given spin configuration predicted whether it constituted an ordered or disordered phase.

In the linear regression problem, we assumed no prior knowledge about the origin of the dataset, i.e. an Ising model with pairwise interactions between every spin pair was used. This generalized model gave rise to that OLS and Ridge regression learned nearly symmetric weights $J = -0.5$. Lasso regression tends to break this symmetry and perfectly determine the coupling $J = -1$. With regularization $\lambda = 10^{-2}$, Lasso gave an $R^2$ score of 1. For the neural network, this was a challenging set of features to predict from, since most of them lack any explanatory ability. The network was shown to neglect the non-contributing features and recreate the energy, given that it was sufficiently penalized. The best model achieved an $R^2$ score of 0.933.

Logistic regression was not able to correctly fit the two-dimensional Ising model as it is not a linear model, and was, in the case with both Gradient Descent and Newton-Raphson as optimization methods, no better than just guessing the phase. Neural networks were also used in predicting phases in the 2D Ising model. Our models rivaled those of Metha et. al [1], achieving an accuracy score of 100% on ordered/disordered test data, as opposed to 99.9% . However, ours did not generalize as good on the critical phases, only achieving a score of 76.5%, as opposed to 95.9%.

# Contents

# 1. Introduction

In this project, we will apply several statistical learning methods to data from both the one-dimensional and two-dimensional Ising model. The Ising model is a binary value system, and is in statistical mechanics used to model interacting dipole moments of atomic spins that are arranged in a lattice with $L$ spin sites. In the one-dimensional case, the linear regression methods OLS, Ridge and Lasso will be used to determine the coupling constant for the Ising model Hamiltonian. A multilayer neural network will also applied on the regression problem. The two-dimensional Ising model exhibits a phase transition from a magnetic phase to a phase with zero magnetization at a given critical temperature, called the Curie temperature $T_C$. Both logistic regression and neural networks will be used in order to classify the phase of the two-dimensional Ising model.

In the linear regression problem, we assume no prior knowledge about the origin of the data set. Hence, a Ising model with pairwise interactions between every pair of variables will be used. How this generalization plays out with the linear regression methods will be studied, and the performance metrics MSE and $R^2$ score will be used to assess the models. Learning the Ising Hamiltonian will be done with both linear regression and a neural network.

The goal with the data from the two-dimensional Ising model, which exhibits a phase transition from an ordered phase to a disordered phase at the Curie temperature, is to build a model which will take in a spin configuration and predict whether it constitutes an ordered or disordered phase. We will here use a dataset provided by Metha et. al [1], which consists of ordered, disordered and "critical-like" phases. The ordered and disordered states will be used to train the logistic regressor and neural network. The remaining critical states will be used as a held-out validation set which the hope is that we will be able to make good extrapolated predictions on. If successful, this may be a viable method to locate the position of the critical point in other complicated models with no known exact analytical solution.

This project is structured by first presenting a theoretical overview of the Ising model and the aforementioned statistical learning methods in Section 2. This is followed by a presentation on the approach to study the various computations of interest in Section 3. Next, the results of the implementation are presented and discussed in Section 4, before subsequently they are concluded upon in Section 5. Lastly, an outline of possible continuations of the models, with respect to the implementation, are presented in Section 6.

# 2. Theory

## 2.1. The Ising Model

The Ising model is a binary value system where the variables of the model can take two values only, e.g. $\pm 1$ or 0 and 1. In statistical mechanics, the Ising model is used to model interacting magnetic dipole moments of atomic spins. The spins, denoted by $s$, are are arranged in a lattice with $L$ spin sites, allowing each spin to interact with its neighbors. The two dimensional square lattice Ising model exhibits a phase transition from a magnetic phase(a system with finite magnetic moment) to a phase with zero magnetization at a given critical temperature, called the Curie temperature, $T_C$ [2]. The Curie temperature is

$$T_C = \frac{2}{\log\left(1+\sqrt{2}\right)} \approx 2.269 \text{ in energy units}, \tag{2.1}$$

as shown by Lars Onsager [3].

By simplifying the interaction between the spins to nearest neighbors only and assuming that the nearest neighbors have the same interaction strength, the Hamiltonian of the system can be defined as

$$H = -J \sum_{\langle kl \rangle} s_k s_l, \tag{2.2}$$

where the spins $s = \pm 1$, $\langle kl \rangle$ indicates that the sum is over the nearest neighbors only, $J$ is a coupling constant expressing the strength of the interaction between neighboring spins.

For $J > 0$ it is energetically favorable for neighboring spins to be aligned, and the magnetic system will have a ferromagnetic ordering. Ferromagnetic materials exhibit a long-range ordering phenomenon where a given magnetic moment, through interactions between nearest neighbors, can influence the alignment of spins that are separated from the given spin by a macroscopic distance. At temperatures below the Curie temperature, $T_C$, the ordering of spin states lead to a phenomenon called spontaneous magnetization. That is, the lattice has a net magnetization even in the absence of an external magnetic field. For temperatures $T \geq T_C$, however, the ferromagnetic property disappears as a result of thermal agitation. [2]

In the linear regression part of this study, we will consider the one-dimensional Ising model with nearest neighbor interactions on a chain of length $N$ with periodic boundary conditions, that is, spins at the boundary will have its nearest neighbors at the opposite boundary. In one-dimension Equation 2.2 reduces to

$$H = -J \sum_{j=1}^{N} s_j s_{j+1}, \tag{2.3}$$

This model does not exhibit any phase transition at finite temperature.

## 2.2. Linear Regression (revisited)

Linear regression is used to predict the value of a continuous dependent variable by simply computing a weighted sum of the input features. A study of three regression methods, namely, Ordinary Least Squares (OLS), Ridge, and Least Absolute Shrinkage and Selection Operator (Lasso), were carried out by the authors in [4]. For a detailed discussion of the aforementioned regression methods, as well as the mean squared error (MSE), coefficient of determination ($R^2$ score), bias and variance, please consult Appendix A or [4].

## 2.3. Logistic Regression

Some regression algorithms can be used for classification. Logistic regression can be used to classify instances into different classes by modelling the probability of a certain class or event existing. Unlike linear regression, the dependent variables (also called responses or outcomes), $y_i$, is categorical, that is, $y_i$ are discrete and only take values from $K = 0, ..., K-1$ (i.e. $K$ classes). The goal is to predict the output classes from the design matrix $X \in \mathbb{R}^{n \times p}$ made of $n$ samples, each of which carries $p$ features or predictors, with the primary goal being able to identify which classes unseen samples belong to. [5] [6]

We limit the following discussion to logistic regression as a binary classifier only, i.e. only two classes, here called positive and negative with corresponding outputs $y_i = 1$ and $y_i = 0$.

### 2.3.1. The Sigmoid Function

In linear regression the output is the weighted sum of inputs. In logistic regression the output is not the weighted sum directly, we rather pass it through an activation function that can map any real value between 0 and 1. [6] The activation function is known as the sigmoid (or logit) function, and is given by

$$\sigma(t) = \frac{1}{1+e^{-t}} = \frac{e^t}{1+e^t} \tag{2.4}$$

Note that $1 - \sigma(t) = \sigma(-t)$

The sigmoid function is shown in Figure 2.1.

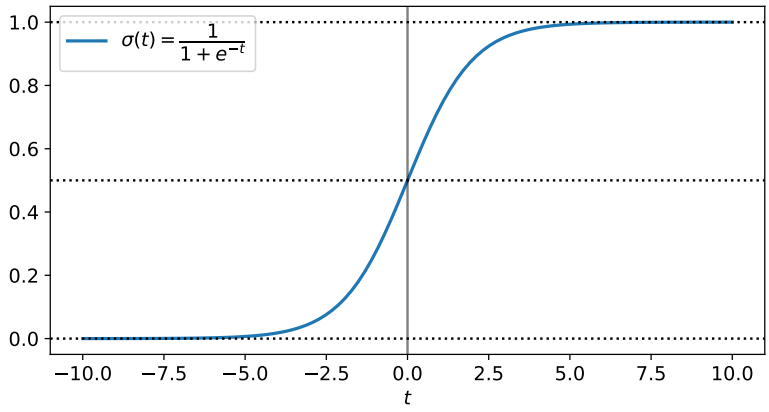



**Figure 2.1:** *The sigmoid function is an activation function that can map any real value between 0 and 1.*

As seen in the figure, the value of the sigmoid function always lies between 0 and 1. The value is exactly 0.5 at $t = 0$. Thus, 0.5 can be used as the probability threshold, $p$, to determine the classes. If $p \geq 0.5$ the instance $t$ belongs to the positive class ($y = 1$), or else we classify it as the negative class ($y = 0$). [6]

### 2.3.2. The Logistic Regression Model

A linear regression model can be represented by

$$\boldsymbol{y} = \boldsymbol{\beta}^T \boldsymbol{X} \tag{2.5}$$

The logistic regression models estimated probability is then

$$\hat{p} = \sigma(\boldsymbol{\beta}^T \boldsymbol{X}), \tag{2.6}$$

and the logistic regression model prediction is hence

$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < 0.5 \\ 1 & \text{if } \hat{p} \geq 0.5 \end{cases} \tag{2.7}$$

### 2.3.3. The Cost Function

Like for linear regression, we define a cost for our model and the objective will be to minimize the cost.

The cost function of a single training instance can be given by

$$C(\boldsymbol{\beta}) = \begin{cases} -\log(\hat{p}) & \text{if } y = 1 \\ -\log(1-\hat{p}) & \text{if } y = 0 \end{cases} \tag{2.8}$$

Figure 2.2 shows that this cost function makes sense.

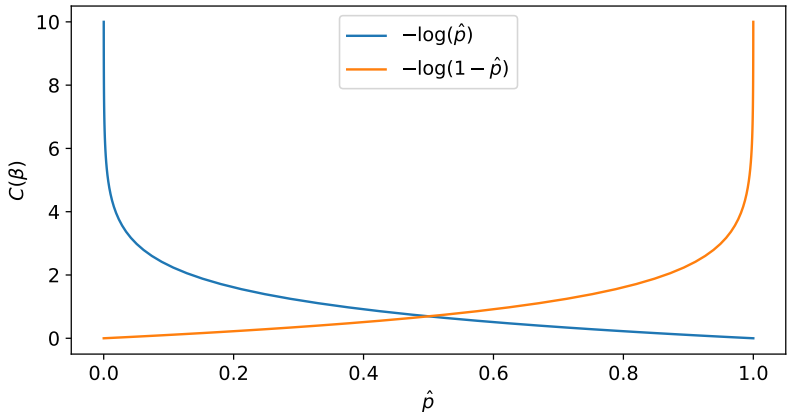



**Figure 2.2:** *The logistic cost function for a single training instance given by Equation 2.8.*

The objective of the cost function is to heavily penalize the model if it predicts a negative class ($y = 0$) if the actual class is positive ($y = 1$) and vice-versa. [6]

As seen in the above figure, for $C(\beta) = -\log(\hat{p})$ the cost nears 0 as $\hat{p}$ approaches 1 and as $\hat{p}$ nears 0 the cost goes toward infinity (that is, we penalize the model heavily). Similarly, for $C(\beta) = -\log(1-\hat{p})$, when the actual value is 0 and the model predicts 0, the cost is 0, and the cost goes toward infinity as $\hat{p}$ approaches 1.

The total likelihood for all possible outcomes from a dataset $\mathcal{D} = \{(y_i, x_i)\}$ with binary labels $y_i \in \{0, 1\}$ and where the data points are drawn independently, can be defined in terms of the product of the individual probabilities of a specific outcome $y_i$ [5]:

$$\hat{P}(\mathcal{D}|\boldsymbol{\beta}) = \prod_i (\hat{p}(y_i = 1|x_i\boldsymbol{\beta}))^{y_i}(1 - \hat{p}(y_i = 1|x_i\boldsymbol{\beta}))^{1-y_i} = \prod_i \hat{p}_i^{y_i}(1-\hat{p}_i)^{1-y_i}, \tag{2.9}$$

which should be maximized by the logistic regressor. Using the Maximum Likelihood Estimation (MLE) principle, we obtain the log-likelihood and the cost function over the whole dataset becomes, by arranging the sign so that it becomes a minimization problem,

$$J(\boldsymbol{\beta}) = -\sum_{i=1}^{n} \left(y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i)\right) \tag{2.10}$$

Equation 2.10 is known in statistics as the cross-entropy.

The gradient of the cross-entropy with regard to the model parameters $\boldsymbol{\beta}$ in compact matrix notation is given by

$$\nabla_\beta J = -\boldsymbol{X}^T(\boldsymbol{y} - \hat{\boldsymbol{p}}) \tag{2.11}$$

in [5].

There is no closed-form solution to compute the value of $\boldsymbol{\beta}$ that minimizes the cross-entropy, i.e. the equation $\nabla_\beta J = \mathbf{0}$ has no analytic solution. The cross-entropy is, however, convex, so Gradient Descent (or any other optimization algorithm) is guaranteed to find the global minimum (with the right learning rate and sufficient time) [6].

As with linear regression, we often supplement the logistic regressor with additional regularization terms to the cross-entropy, usually $L_1$ and $L_2$ regularization [5]. With $L_2$ regularization the cross entropy reads

$$J(\boldsymbol{\beta}) = -\sum_{i=1}^{n} \left(y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i)\right) + \lambda \|\boldsymbol{\beta}\|_2^2, \tag{2.12}$$

which simply adds a term to the gradient

$$\nabla_\beta J = -\boldsymbol{X}^T(\boldsymbol{y} - \hat{\boldsymbol{p}}) + 2\lambda\boldsymbol{\beta} \tag{2.13}$$

where $\lambda$ is the regularization parameter.

### 2.3.4. Accuracy Score

The performance of a logistic regression model can be measured with the accuracy score, given by

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} I(t_i = y_i)}{n}, \tag{2.14}$$

where $I$ is the indicator function, 1 if $t_i = y_i$ and 0 otherwise, where $t_i$ represents the target and $y_i$ the outputs. A perfect classifier will have a score of 1.

### 2.3.5. Receiver Operating Characteristic

Another metric for assessing the quality of the model, is the receiver operating characteristic (ROC) curve, which tells us how well the model correctly classifies the different labels. The ROC curve is created by plotting the true positive rate (the rate of predicted positive classes that are positive) against the false positive rate (the rate of predicted positive classes that are negative) for varying thresholds.

An estimate of how well the model is performing can be obtained by computing the area under the curve (AUC) of the ROC curve. Pure guessing will get an AUC of 0.5. A perfect score will get an AUC of 1.0. [7]

## 2.4. Optimization and Gradient Methods

### 2.4.1. Gradient Descent

Let $f\colon U \to \mathbb{R}$ be a function defined on a subset of $\mathbb{R}^n$. Suppose that $x \in U$ and that $f$ is differentiable in $x_0$. Then from the point $x_0$, the function $f$ has the steepest decrease in the direction along $-\nabla f(x_0)$. Hence, unless $x_0$ gives a minimum for $f$, given $\gamma_0 > 0$ small enough, we have that for $x_1 = x_0 - \gamma_0 \nabla f(x_0)$, the inequality $f(x_1) \leq f(x_0)$ holds. For all $n \in \mathbb{N}$, define

$$x_n = x_{n-1} - \gamma_{n-1} \nabla f(x_{n-1}),$$

where $\gamma_{n-1} > 0$ is some number. If the $\gamma_n$ are chosen sufficiently small, the sequence $\{x_n\}_{n\in\mathbb{N}}$ converges to a local minimum. Larger $\gamma_n$ make the convergence faster.

### 2.4.2. Newton-Raphson method

This section is based on [8, Section 5.6].

Let $A \subseteq \mathbb{R}^n$ and suppose that $f\colon A \to \mathbb{R}^n$ is a differentiable function. If $x_0 \in A$, then for $x \in A$ "close" to $x_0$, the function $x \mapsto f(x_0) + Df(x_0)(x - x_0)$ is a good approximation of $f$. Assume that $f$ has a zero close to $x_0$. To try and find this zero, instead of solving $f(x) = 0$, we solve the system of linear equations given by

$$f(x_0) + Df(x_0)(x - x_0) = 0.$$

If $Df(x_0)$ is invertible, we get the solution

$$x_1 = x_0 - (Df(x_0))^{-1} f(x_0).$$

We can hope that $x_1$ is even closer than $x_0$ to the actual zero of $f$. Repeating the process with $x_1$ instead of $x_0$, we find a new approximation

$$x_2 = x_1 - (Df(x_1))^{-1} f(x_1).$$

If we continue this for all $n \in \mathbb{N}$, we get a sequence $\{x_n\}_{n\in\mathbb{N}}$ where

$$x_n = x_{n-1} - (Df(x_{n-1}))^{-1} f(x_{n-1}).$$

Hopefully the sequence converges to a point $x = \lim_{n\to\infty}$ such that $f(x) = 0$. This root-finding algorithm is called the *Newton-Raphson method*.

In general Newton-Raphson is not guaranteed to work. Sufficient conditions are given in the Kantorovich theorem:

**Theorem 2.1.** *Let $f\colon U \to \mathbb{R}^n$ be a differentiable function defined on an open convex subset $U$ of $\mathbb{R}^n$. Assume that the Jacobian $Df$ of $f$ is Lipschitz continuous on $U$, i.e. that there exists an element $M \in \mathbb{R}$ such that*

$$\|Df(x) - Df(y)\| \leq M\|x - y\|$$

*for any $x, y \in U$.*

*Let $x_0 \in U$ and suppose that $Df(x_0)$ is invertible. Find a number $K$ with $\|Df(x_0)^{-1}\| \leq K$, and assume that the closed ball $\overline{B}\left(x_0, \frac{1}{KM}\right)$, with center in $x_0$ and radius $\frac{1}{KM}$ is contained in $U$. If*

$$\|x_1 - x_0\| = \left\|Df(x_0)^{-1} f(x_0)\right\| \leq \frac{1}{2KM},$$

*then $Df(x)$ is invertible for all $x$ in the open ball $B\left(x, \frac{1}{KM}\right)$. If we start Newton-Raphson in $x_0$, then all the points $x_n$ are contained in $B\left(x_0, \frac{1}{KM}\right)$ and the limit $x = \lim_{n\to\infty} x_n$ exists and satisfies $f(x) = 0$.*

## 2.5. Neural Network

Neural networks comprise a huge family of flexible function suitable of approximating nearly any function. Originally used as a simple model of the brain by McCulloch and Pitts in 1943 [9], the field of neural networks has proved to be extremely useful in the world of modelling and machine learning. Today, the progress is still going strong, with application in a vast number of fields and many different flavors of neural networks

### 2.5.1. Architecture of the Feed-Forward Neural Network

In this project, we focus on the feed-forward neural network, hereafter referred to simply as neural network. A neural network is in essence a real-valued function that may take any number of inputs and outputs. It's behaviour is completely determined by its architecture, which is comprised of hidden layers.

A hidden layers is a set of nodes that are wired to the previous hidden layer, or the inputs, in case it is the first hidden layer in the network. For the i'th node in the first hidden layer, its activation is calculated as

$$z_j^1 = \sum_{i=1}^{N} W_{ji}^1 x_j + b_j^1, \tag{2.15}$$

where $N$ is the number of inputs $x$, $W^1$ is a matrix of the weights connecting the inputs to the first hidden layer, and $b^1$ is a constant bias term added to the activation. Taking the activation as the output of the first hidden layer, Equation 2.15 defines a method of feeding information to the next hidden layer as well, and ultimately through the entire network. However, it can easily be seen from the equation that the feeding of information from one layer to another is a affine transformation. Since a affine transformation of a affine transformation is a affine transformation in itself, adding more layers to a network fails in making a more complex model. To avoid this collapse of layers, we need to introduce a non-linear function $f^1$ to the activation before passing it to the next layer

$$a_j^1 = f^1(z_j^1) \tag{2.16}$$

Non-linear functions used are typically sigmoid, tanh or the Relu, and enables neural network of becoming powerful and flexible models.

### 2.5.2. Backward Propagation

To evaluate how good a neural network performs on data, some kind of cost function is consulted

$$C(y, \tilde{y}) = \sum_{i=1}^{N} c(y_i, \tilde{y}_i) \tag{2.17}$$

Where the sum is over $N$ data-points, $y_i$ is the target of the i'th data point, and $\tilde{y}_i$ is the accompanying prediction of the network given by $\tilde{y}_i = NN(X_i)$. $c$ is typically the square loss for regression

$$c(y_i, \tilde{y}_i) = (y_i - \tilde{y}_i)^2, \tag{2.18}$$

and cross entropy for binary classification

$$c(y_i, \tilde{y}_i) = y_i \log(\tilde{y}_i) + (1 - y_i) \log(1 - \tilde{y}_i), \tag{2.19}$$

For an untrained neural network, the weights $W$ connecting all the layers are typically initialized in random fashion, popularly as $W \sim N(0,1)$, and the biases $b$ set to a small number such as 0.01.

To improve the model from the initial random state, we would like to minimize the cost function with respect to the weights. As the neural network is a very complex function, and not even a convex problem where all minima are global minima, the optimal weights can not be found by $\frac{dC}{dW_{ij}^l} = 0$. Instead, we resort to using gradient decent methods.

To compute the gradient $\frac{dC}{dW_{ij}^l}$, we use the chain rule repeatedly to produce an expression that keeps track of how the variation of the cost function propagates backward in the neural network, so-called backward propagation.

To start, we calculate the derivative of the cost function with respect to the weights of the last hidden layer:

$$\frac{dC}{dw_{ij}^L} = \frac{dC}{da_j^L}\frac{da_j^L}{dw_{ij}^L} = \frac{dC}{da_j^L}f'(z_j^L)\frac{dz_j^L}{dw_{ij}^L} = \frac{dC}{da_j^L}f'(z_j^L)a_i^{L-1} \tag{2.20}$$

where we have used the chain rule repeatedly and the definition of the activation Equation 2.15.

Defining $\delta_j^L = \frac{dC}{da_j^L}f'(z_j) = \frac{dC}{dz_j^L}$, we can now look at the derivative of an arbitrary layer $l$:

$$\delta_j^l = \frac{dC}{dz_j^l} = \sum_k \frac{dC}{dz_k^{l+1}}\frac{dz_k^{l+1}}{dz_j^l} = \sum_k \delta_k^{l+1}\frac{dz_k^{l+1}}{dz_j^l} = \sum_k \delta_k^{l+1} w_{kj}^{l+1} f'(z_j^l) \tag{2.21}$$

This enables us to calculate the error propagation to an arbitrary layer, and we can calculate the gradient from Equation 2.20. The update rules for a standard GD thus becomes

$$\begin{aligned} w_{jk}^l &\to w_{jk}^l - \mu\delta_j^l a_k^{l-1} \\ b_j^l &\to b_j^l - \mu\delta_j^l \end{aligned} \tag{2.22}$$

where $\mu$ is the learning rate.

### 2.5.3. Regularization

Neural networks quickly grow in complexity as you increase the number of hidden layers and number of neurons for each layer. One must take care not to overfit the networks. A common approach is to add the L2-norm of the weights to the cost function as a form of penalty, in the spirit of Ridge Regression:

$$C(y,\tilde{y}) = \sum_i (y_i - \tilde{y}_i)^2 + 0.5\lambda \sum_{ijl} {w_{ij}^l}^2$$

This changes the update rule for the weights to

$$w_{jk}^l \to w_{jk}^l - (\mu\delta_j^l a_k^{l-1} + \lambda' w_{jk}^l) \tag{2.23}$$

where $\lambda' = \mu\lambda$.

# 3. Method

## 3.1. Learning the Ising Hamiltonian with Linear Regression

We want to predict the total energy of the system based on the spin configuration. Assuming no knowledge about the interactions between the spins, a choice of model can be

$$H_{\text{model}}(\mathbf{s}^i) = \sum_{j=1}^{N}\sum_{k=1}^{N} J_{j,k} s_j^i s_k^i, \tag{3.1}$$

where $\mathbf{s}^i$ is a particular spin configuration. By collecting all the two-body interactions $\{s_j^i s_k^i\}$ for all $i = 1, \ldots, N$ in a design matrix $\mathbf{X}$, and all the non-local coupling strengths $J_{j,k}$ in a vector $\mathbf{J}$, we can write this as

$$(H_{\text{model}}(\mathbf{s}^1), \ldots, H_{\text{model}}(\mathbf{s}^N))^T = \mathbf{X}\mathbf{J}.$$

With this formulation we can use regression methods from project 1 (OLS, Ridge and Lasso), see Appendix A or [4], to find the coefficients $\mathbf{J}$.

We generate $N = 10\,000$ spin configurations, each consisting of 40 spins. The spins take values in $\{-1, 1\}$ with a uniform distribution. The energy of each system is calculated in accordance with Equation (2.3) for $J = 1$.

In order to assess the models, we use the $R^2$ score, given by Equation A.9, and mean squared error (MSE), given by Equation A.8, as measures of performance.

A bias-variance analysis using the bootstrap resampling method for best model selection will also be carried out.

## 3.2. Identifying 2D Ising Model Phases with Logistic Regression

Here, we use the binary logistic regression, as discussed in Section 2.3, and the two-dimensional Ising model with Hamiltonian given by Equation 2.2. Opposed to the one-dimensional Ising model, we will get a phase transition from an ordered phase to a disordered phase at the Curie temperature, given by Equation 2.1. Logistic regression will be used as a binary classification method to predict if a given state of the two-dimensional Ising model is ordered or disordered. We will be looking at a system of $L = 40$ spins in each dimension, that is, a total $L^2 = 1\,600$ spins. The data, provided by Metha et. al [1], is a set of $10\,000$ spin configurations (with $40 \times 40$ spins each) labeled ordered 1 or disordered 0. The goal is to build a model which will take in a spin configuration and predict whether it constitutes an ordered or disordered phase.

The lattices are represented as flattened arrays with $1\,600$ elements instead of a matrix of $40 \times 40$ elements. The dataset is first divided into three pieces, each consisting of either the ordered, disordered or "critical-like" phase. We use both ordered and disordered states to train the logistic regressor. We will estimate the test error on unseen ordered and disordered states, to assess the regularization parameter. The remaining critical states will be used as held-out validation set on which the hope is that we will be able to make good extrapolated predictions. If successful, this may be a viable method to locate the position of the critical point in other complicated models with no known exact analytical solution [1].

Figure 3.1 shows some states from the dataset for both the ordered and disordered phase, and the critical region as well.

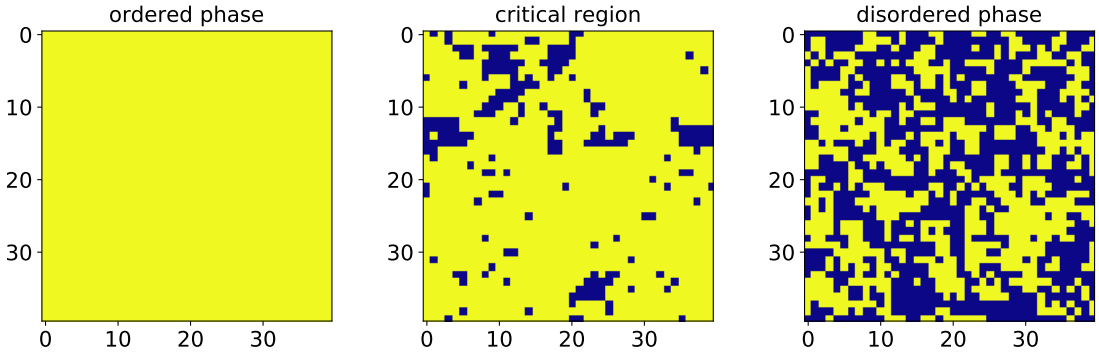



**Figure 3.1:** *Examples of typical states of the two-dimensional Ising model for three different temperatures in the ordered phase, the critical region and the disordered phase. The system dimension is $L = 40$ spin sites.*

In order to minimize the cross entropy, given by Equation 2.10, we will implement both Gradient Descent (see Section 2.4.1) and Newton-Raphson (see Section 2.4.2).

We first attempt doing the logistic regression using the Gradient Descent method for optimization. The data set is quite large and slow to work with, so we begin with a small training set. We try different values for both the learning rate $\eta$ and the regularization parameter $\lambda$ in hope to find something that works out, as measured by the accuracy score, see Equation 2.14. Afterwards, maybe affected by hubris, we try to use these parameters on a much larger training set, with the hope that we can achieve an even higher prediction accuracy. We evaluate the model by the ROC curve for the model performance on the critical phase ising states.

To conclude the logistic regression analysis, after seeing the Gradient Descent's poor optimization performance, we try to use the Newton-Raphson method (see Section 2.4.2). This method has a notoriously slow iterative scheme, so we will only apply it on a small training set and with few, but sufficiently many, iterations. A ROC curve will also be presented for this method.

## 3.3. Neural Networks

### 3.3.1. Implementation

For this project, neural networks has been implemented as a python class using standard numpy for linear operations. The architecture can be chosen arbitrarily, i.e. the number of neurons, layers, activation functions and a cost function .

To train the network, one must supply a training set and validation set. The choice of optimization is batch gradient decent. The learning rate, penalty, batch size and number of epoch must be supplied. After each epoch, the accuracy of the network is evaluated and stored using the cost function on both the training data and validation data. This is then used later to evaluate how the network learned.

### 3.3.2. Regression on Energy of Generalized Ising Model using Neural Networks

In the same manner as described in Section 3.1, a set of energies as targets were produced by the real Ising model Equation 2.3, but the features were assumed to originate from an unknown model(generalized Ising model). Using a neural network, the energies were attempted learned

from the 1600 features, of which only 80 have real explanatory ability(those who contribute in the real model). The parameters of the neural network can be summarized in Table 3.1.

**Table 3.1:** *Parameters of the neural network.*

| | |
|---|---|
| Number of Neurons | 400 (one layer) |
| Hidden layer activation | tanh |
| Output activation | Unmodified |
| Cost function | Squared Loss |
| Learning rate | $7 \cdot 10^{-5}$, $9 \cdot 10^{-5}$, $11 \cdot 10{-}5$ |
| Penalty | $5 \cdot 10{-}4$, $1 \cdot 10^{-3}$, $2 \cdot 10^{-3}$ |
| Batch size | 100 |
| Epochs | 50 |

Several models are trained using a grid search using the different learning rates and penalties. The R2-score is used to evaluate the models.

To investigate how the networks respond to the dummy features introduced by the generalized Ising model, we introduce a measure "Connection Strength". The connection strength (CS) of a neuron $n_j^l$ is defined as the absolute sum of all the weights that connect that neuron to the next layer:

$$CS = \sum_i |w_{ji}^{l+1}| \tag{3.2}$$

This is may be though of a measure of how much a particular neuron connects to the next layer. When evaluated on a input feature, we interpret it as how much that feature in particular influence the model. In the case that the CS is zero, the feature has no influence whatsoever.

### 3.3.3. Identifying 2D Ising Model Phases with Neural Networks

The data trained on are the same as described in Section 3.2, from [1]. The network is trained on the combined orders and disordered states. The model is then evaluated on an independent test set collected from the same combined set. In addition, the model is evaluated on a set consisting of the critical states to explore the generalizability of the model.

The parameters of the neural network can be summarized in this table:

| | |
|---|---|
| Number of Neurons | 400, One layer |
| Hidden layer activation | Sigmoid |
| Output activation | Sigmoid |
| Cost function | Cross Entropy |
| Learning rate | $1e-5$, $2e-5$, $3e-5$ |
| Penalty | $1e-5$, $1e-4$, $1e-3$ |
| Batch size | 100 |
| Epochs | 50 |

Several models are trained using a grid search using the different learning rates and penalties. The accuracy score, given by Equation 2.14, is used to evaluate the models.

# 4. Results and Discussion

The programs containing the implementation of the methods discussed in the previous section, as well as the results presented in this section, can be found at the GitHub repository

https://github.com/nicolossus/FYS-STK4155-Project2

The procedures for producing the following results are contained in Jupyter Notebooks found here.

## 4.1. Learning the Ising Hamiltonian with Linear Regression

Accompanying notebook: linreg_ising.ipynb.

Figure 4.1 shows the coupling matrix obtained by performing OLS regression.

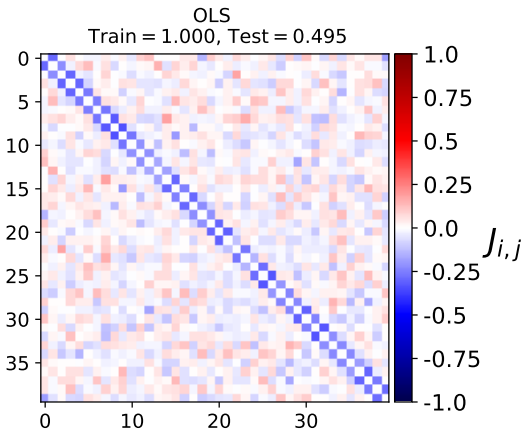



**Figure 4.1:** *Learned coupling matrix $J_{ij}$ for the Ising model ansatz in Equation 3.1 for ordinary least squares (OLS) regression.*

In the linear regression problem, we assumed no prior knowledge about the origin of the data set. Hence, a Ising model with pairwise interactions between every pair of variables is used here. This generalized model gives rise to that OLS learn nearly symmetric weights $J = -0.5$, as seen in Figure 4.1. What the generalization of the model means, is that we included both $s_j s_i$ and $s_i s_j$ for all $i, j$ in the features. Since these quantities are equal, we do not get a unique solution for OLS, so we resort to singular value decomposition (SVD). SVD gives the coefficients with the least $L2$-norm. Hence we tend to get solutions $J_{i,j} = J_{j,i} = -0.5$.

Figure 4.2 shows the coupling matrices obtained by performing Ridge and Lasso regression with different regularization parameter, $\lambda$, values.

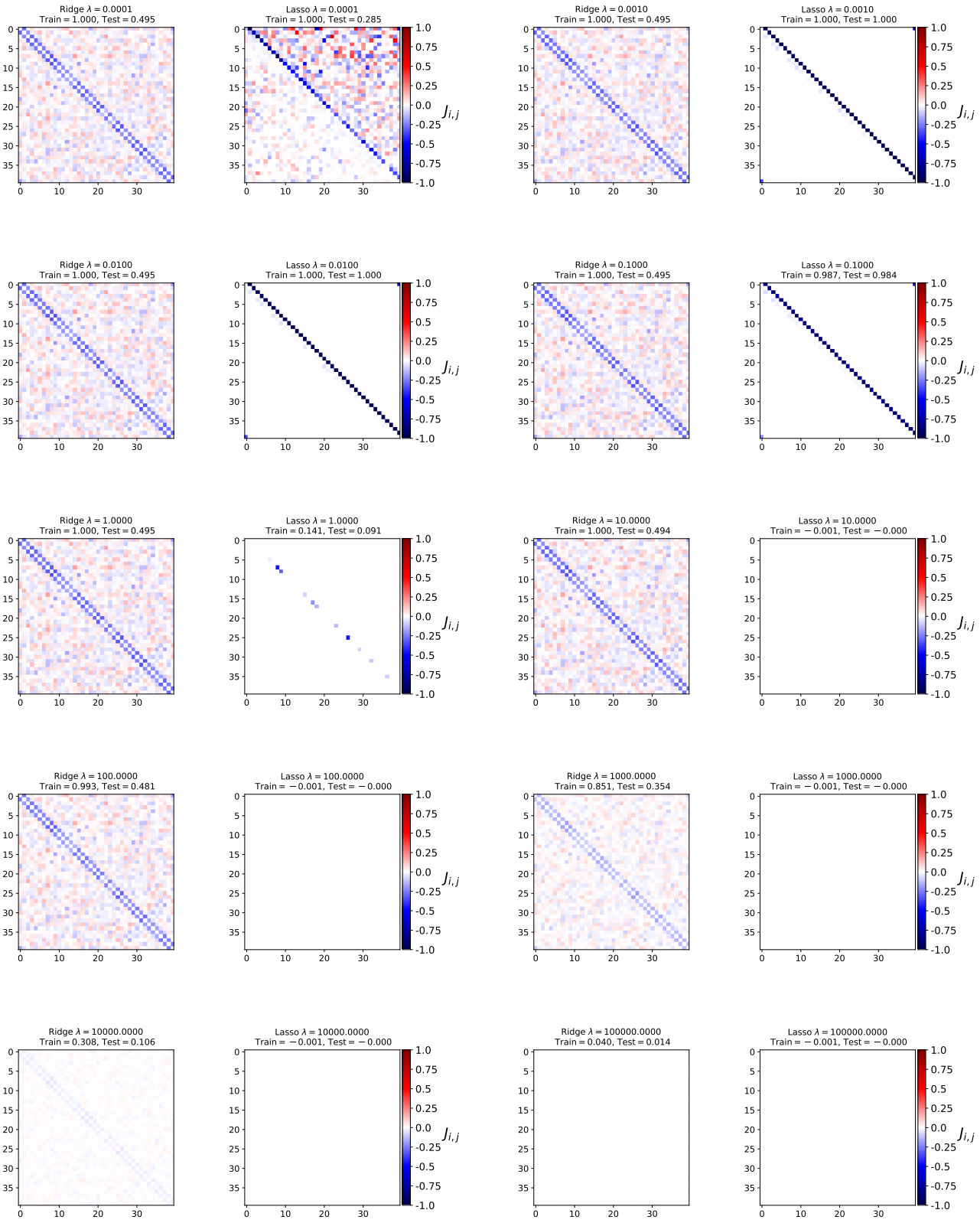
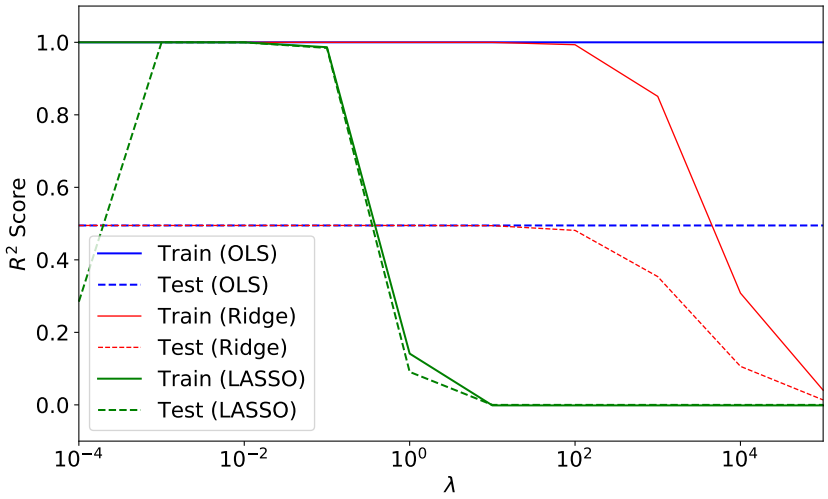



**Figure 4.2:** *Learned coupling matrix $J_{ij}$ for the Ising model ansatz in Equation 3.1 for Ridge regression (to the left in each subfigure) and Lasso regression (to the right in each subfigure) at different regularization strengths $\lambda$.*

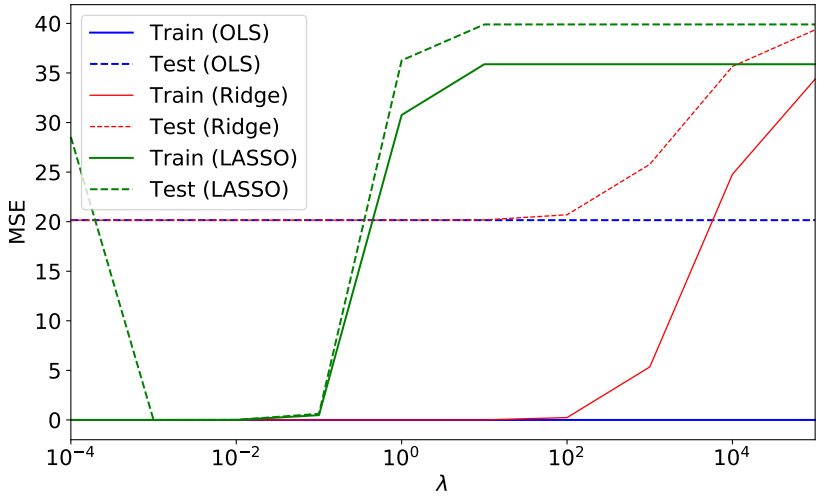
From Figure 4.2 we see that Ridge regression also learn nearly symmetric weights $J = -0.5$. Ridge is similar to the OLS case in this regard, since it uses an $L2$ penalty. Regularization of

Ridge regression does not improve the performance. Lasso regression, however, tends to break this symmetry and perfectly determine the coupling $J = -1$ with proper regularization. With regularization $\lambda = 10^{-2}$, Lasso even gives an $R^2$ score of 1.

Figure 4.3 shows the $R^2$ score and MSE as a function of the regularization parameter $\lambda$.




**(a)**

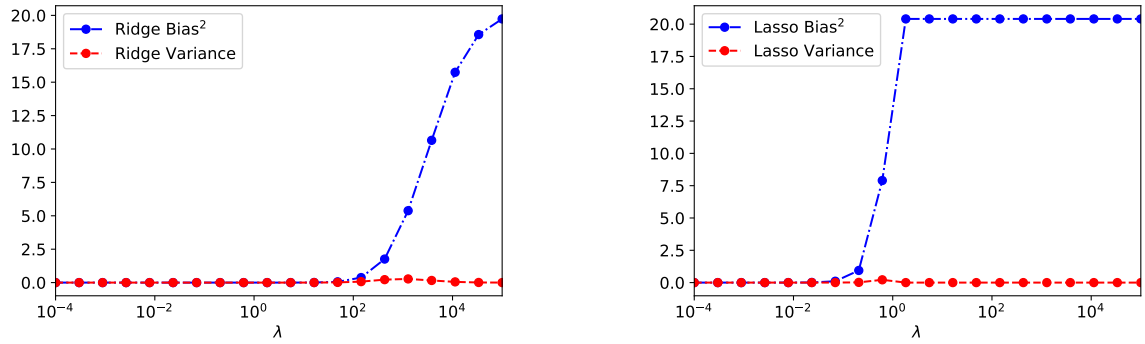



**(b)**

**Figure 4.3:** *Performance of OLS, Ridge and Lasso regression with different regularization strengths on the Ising model as measured by the* **(a)** $R^2$ *score and* **(b)** *MSE.*

From Figure 4.3 it is evident that Lasso with $\lambda = 10^{-2}$ achieves an excellent predictive ability on the test set and by far surpasses the other models for all values of $\lambda$. Using to strong regularization causes the MSE to increase as the model becomes unable to fit the data. It is also noteworthy that OLS and Ridge has notable deviations between the training and test set, which may be an indication of overfitting.

The bias-variance decomposition of Ridge and Lasso regression with different regularization strengths are shown in Figure 4.4.




**(a)** **(b)**

**Figure 4.4:** *Bias-variance decomposition using bootstrap for* ***(a)*** *Ridge regression and* ***(b)*** *Lasso regression with different regularization strengths* $\lambda$.

The bias-variance decomposition corresponds well with the findings from Figure 4.3, which is expected as there is no noise in the data.

## 4.2. Identifying 2D Ising Model Phases with Logistic Regression

Accompanying notebook: logreg_ising.ipynb.

Figure 4.5 shows the accuracy of the logistic regressor with Gradient Descent as optimization method and different values for the learning rate $\eta$ and regularization $\lambda$.

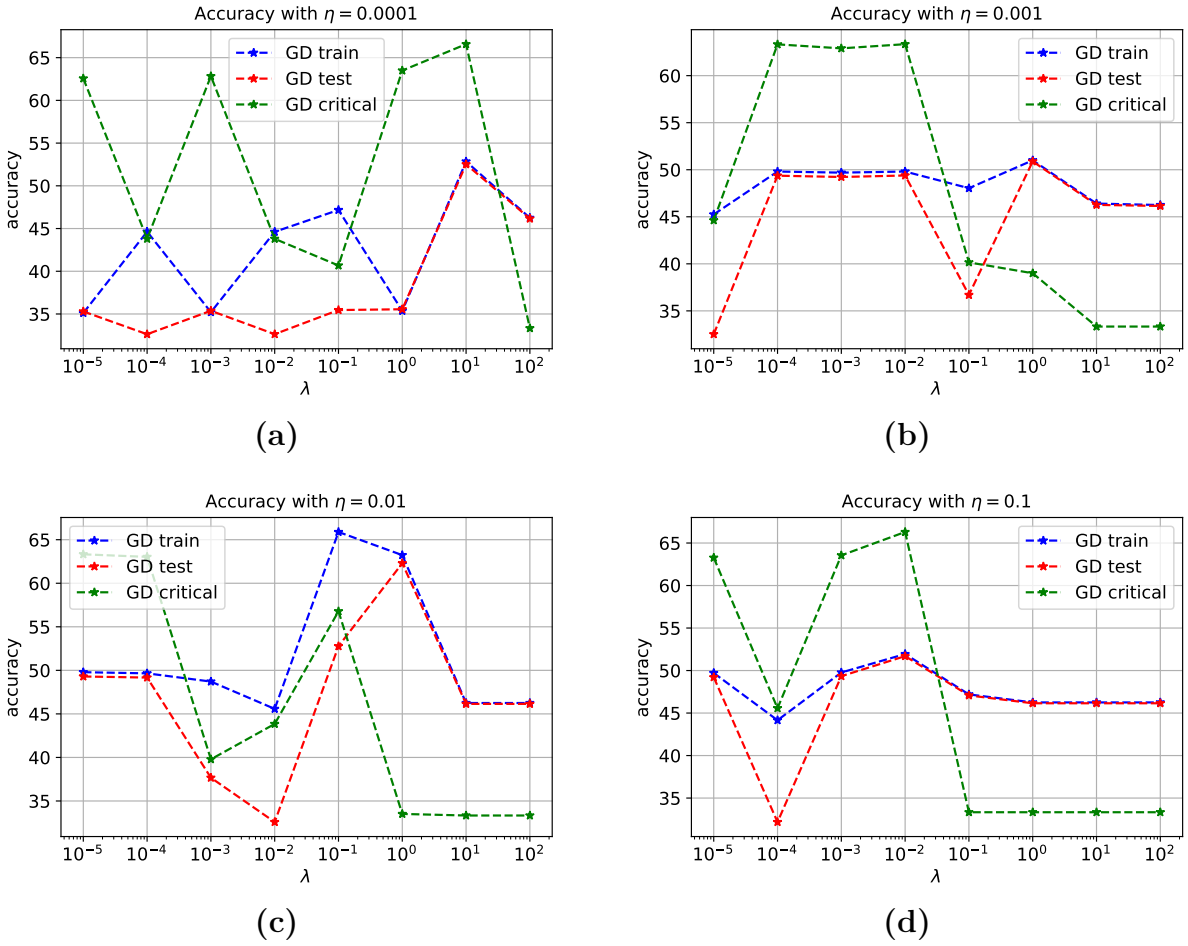



**Figure 4.5:** *The accuracy score of the training, test and critical datasets as a function of the regularization parameter $\lambda$ for different learning rates $\eta$. Gradient descent was used as optimization method.*

In the above figure, we see the accuracies for the training (blue), test (red) and critical (green) datasets. The accuracies fluctuate dramatically, implying that the predictions are all over the place. However, the accuracies seems to be worse with stronger regularization. The difference between test and training sets could in theory tell us about the relative degree of overfitting, but is hard to assess due to the fluctuations. Due to the physics of the Ising model, we expect that predicting the phase close to the critical point will be more difficult. This, however, is not present in our results here, as the regressor seems to just guess for all sets.

There are no clear optimal parameters, and none of the best accuracies are great either. However, we stick with the best parameters for the maximum training accuracy, $\eta = 0.01$ and $\lambda = 0.1$, and try to fit on a larger training set (20%). The result is shown in the ROC curve in Figure 4.6.

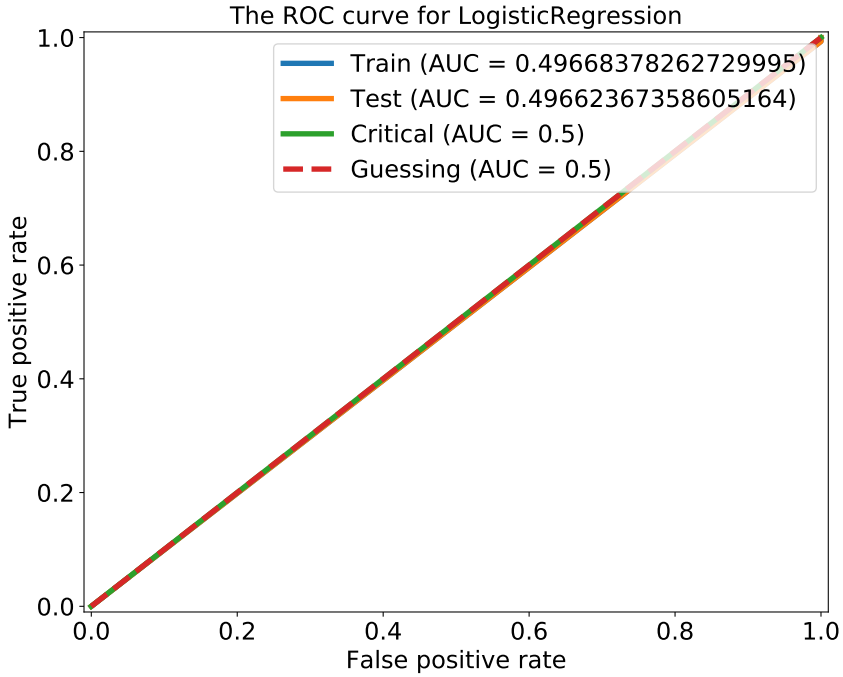



**Figure 4.6:** *ROC curve for our logistic regressor with Gradient Descent as optimization method.*

As seen in the above figure, our logistic regression model with gradient descent does no better than just guessing the phase.

A second optimization method has also been implemented, NR. This method is notoriously slow iterative scheme, so we will only apply it on a small training set and with few iterations. Only 4% training data and 10 iterations

The results from running Newton-Raphson for different $\lambda$ is shown in Table 4.1 and in Figure 4.7. We get a slight top at $\lambda = 10^0 = 1$, so this is our chosen regularization parameter. We plot the ROC curve with this in Figure 4.8 to evaluate the model.

**Table 4.1:** *The logistic regressor's accuracy on the training, test and critical sets with Newton-Raphson as optimization method.*

| | $\lambda$ | Training accuracy | Test accuracy | Critical accuracy |
|---|---|---|---|---|
| 0 | 0.00001 | 95.09615 | 68.80849 | 53.51667 |
| 1 | 0.00010 | 95.09615 | 68.81090 | 53.52333 |
| 2 | 0.00100 | 95.05769 | 68.81330 | 53.51667 |
| 3 | 0.01000 | 94.92308 | 68.86779 | 53.69333 |
| 4 | 0.10000 | 93.59615 | 69.24038 | 54.26667 |
| 5 | 1.00000 | 90.44231 | 70.10497 | 56.50667 |
| 6 | 10.00000 | 86.88462 | 69.99679 | 60.48333 |
| 7 | 100.00000 | 82.38462 | 68.71715 | 64.17333 |

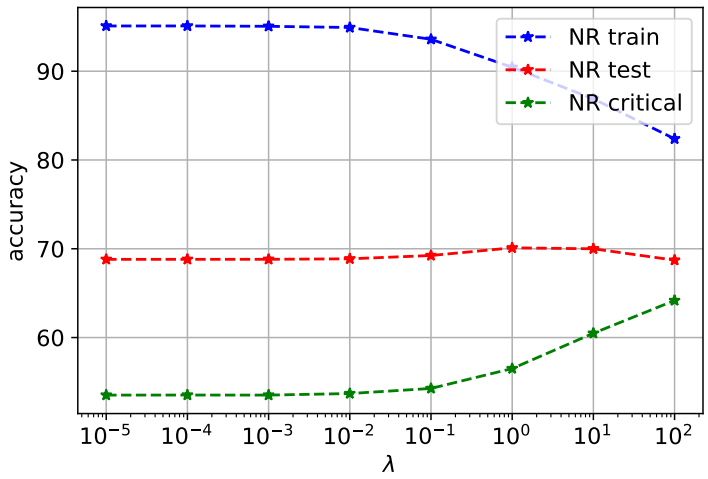



**Figure 4.7:** *The accuracy score of the training, test and critical datasets as a function of the regularization parameter $\lambda$ for different learning rates $\eta$. Newton-Raphson was used as optimization method.*

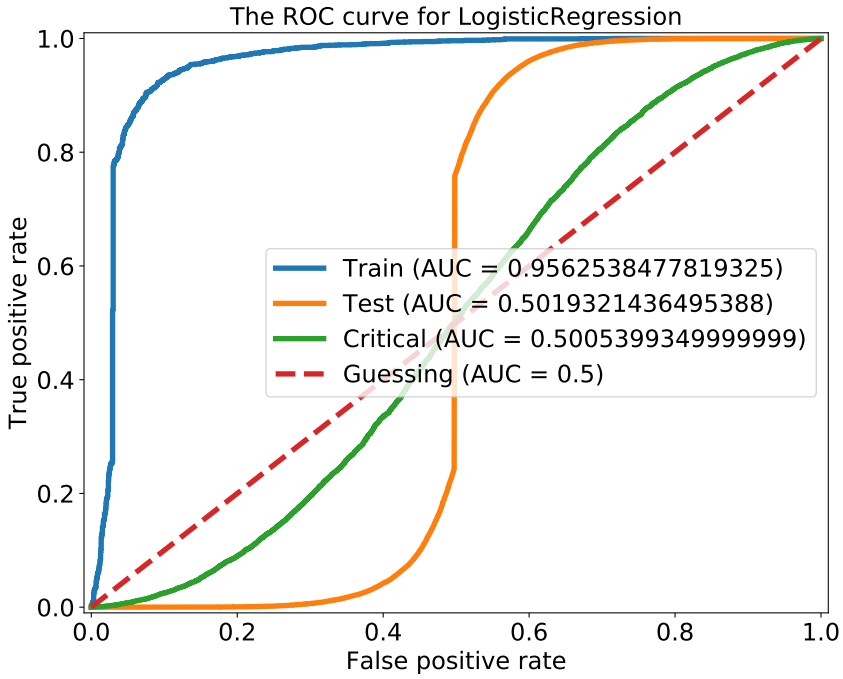


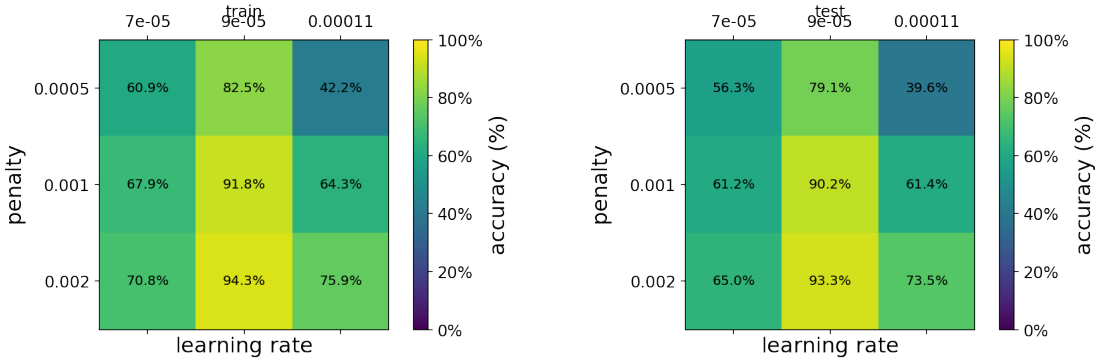

**Figure 4.8:** *ROC curve for our logistic regressor with Newton-Raphson as optimization method.*

As we can see in Fig. 4.8, logistic regression is not able to correctly fit the Ising model. This is not surprising as it is not a linear model. The logistic regression model is no better than just guessing.

## 4.3. Regression on Energy of Generalized Ising Model using Neural Networks

Accompanying notebook: nn_ising.ipynb.

Figure 4.9 shows the R2-score different neural networks predicting the energy of the Ising model, using features produced by the Generalized Ising model.




**(a)** **(b)**

**Figure 4.9:** *R2-score of several neural networks trained and tested on energies produced using the real Ising model Equation 2.3, spin features produced using the generalized Ising model Equation 3.1. All networks have a single hidden layer of 400 neurons, using* tanh *as activation, squared loss as cost function, and L2-regularization. They were trained using a grid search on learning rate and penalty*

As the real Ising model only include local coupling, only 80 of the 1600 features of the generalized model actually contribute to the energy. The interesting question to explore is if the network is able to pick out the useful features and ignore the others. From the above figure, we see that the more heavily penalised models perform generally better. In the same manner that Ridge and Lasso help linear regression suppress features of less importance and reduce variance, the regularizing the network produces better models, presumably by suppressing the weights connecting to the redundant features. To go a step further, a model using $\mu = 9 \cdot 10^{-5}$ and $\lambda = 0.004$ was trained. This model yielded a $R^2$ score of 0.26.

To elaborate the previous point, figure Figure 4.10 visualises the Connection Strength (CS), Equation 3.2, of all the inputs for models using learning rate $\mu = 9 \cdot 10^{-5}$ and penalties 0.0005, 0.001, 0.002 and 0.004.

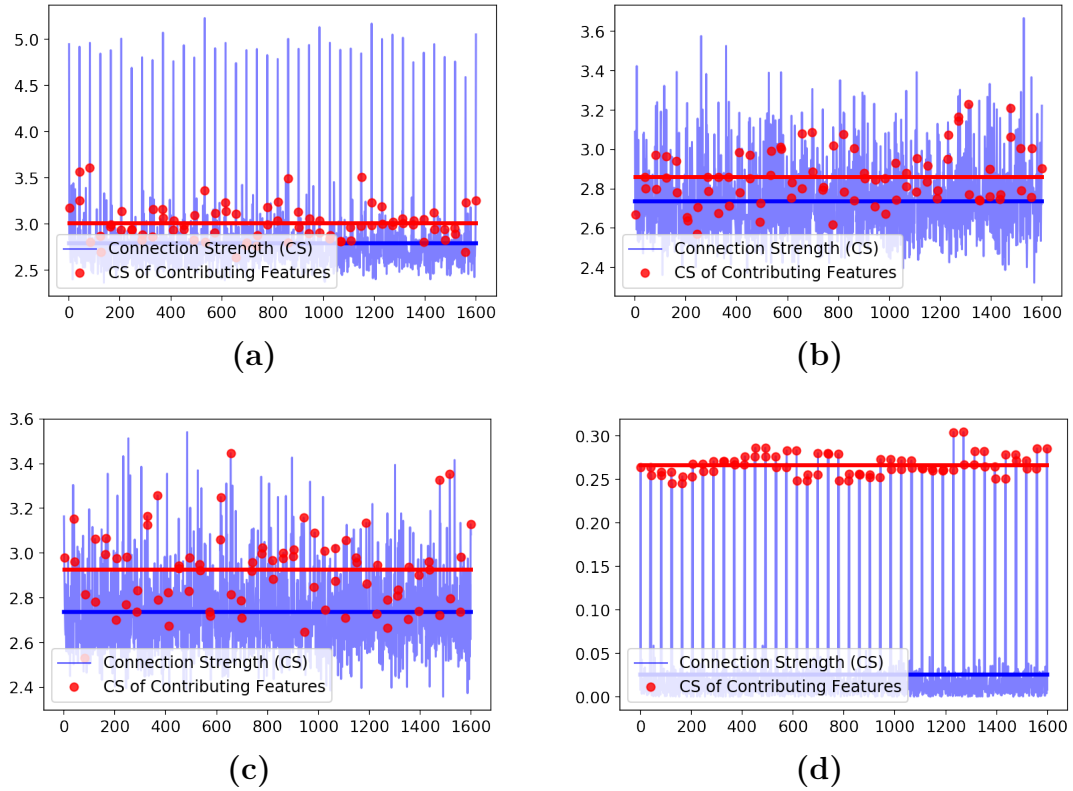



**Figure 4.10:** *Connection strength Equation 3.2 of all inputs for the models of learning rate $\mu = 9e-5$ and penalties* 0.0005, 0.001, 0.002 *and* 0.004. *The red dots highlights the CS of the features that contribute to the energy. The horizontal lines are the average CS of the two groups just described, respectively.*

As suspected, the increase in penalty raises the average CS of the inputs that contribute to the energy, making the network emphasize these inputs more. This ultimately yields a better model, until the penalty is raised to 0.004. From the above figure, the contributing features are separated from the others more than ever. However, the weights have been shrunk to the point where the model is fairly desensitized to the features, with a CS of the order 0.3 instead of 3. Not too surprisingly, the model performed terribly, with a R2-score of 0.26.

As a final note to the exploration of the learning behaviour, figure Figure 4.11 shows the $R^2$ score on validation data progressively as the network is trained.

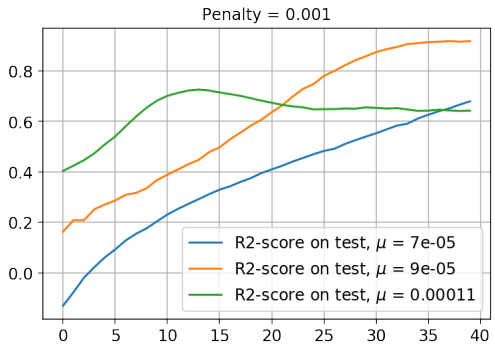



**Figure 4.11:** *R2-score of neural networks evaluated on a independent validation set. The models all use penalty $\lambda = 0.001$, with learning rate $\mu = 7e-5$, $9e-5$ and $1.1e-4$.*

The plot highlights the typical problems of tuning learning rate: The model with the smallest learning rate is constantly improving, but never reaching its potential since the step size is too small. On the other hand, the one with the highest learning rate might be continuously skipping over its local minima after getting fairly close at first, resulting in gradually degrading results.

## 4.4. Identifying 2D Ising Model Phases with Neural Networks

Figure 4.12 shows the accuracy score of several neural networks trained and tested on the combined ordered and disordered set from Mehta et al. in [1]. Additionally, the models were tested on the critical set to explore the generalizability of the models.

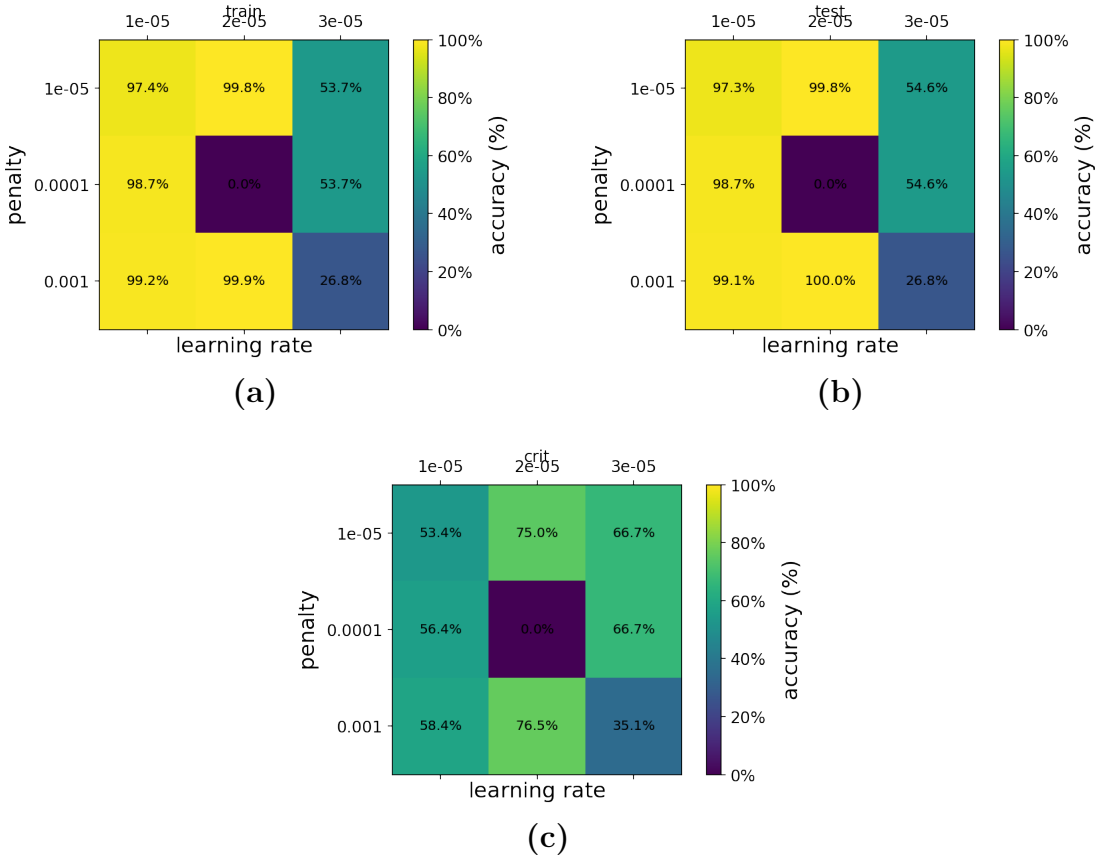



**Figure 4.12:** *Accuracy score of several neural networks trained and tested on the combined ordered and disordered set from Mehta. Additionally, the models were tested on the critical set. All networks have a single hidden layer of 400 neurons, using sigmoid as activation on both the hidden layer and output. They were trained using a grid search on learning rate and penalty(L2-regularization)*

An eye-catching result is the sudden ruination of one of the models, yielding a accuracy score of precisely zero. This is curious, since models with parameters in the same neighborhood yield sensible predictions. When the faulty model was trained, the program once encountered ”value encountered in true divide”. This is a weakness of the cross-entropy. If the network misclassifies a label, and if the network is very sure in its (mis)classification, the derivative of the cross-entropy becomes enormous. This will likely ruin the weights during the next GD step, and the model too.

The fluke aside, our model attained a maximum accuracy score of 100% for the ordered/disordered data meant for testing, rivaling Mehta's model of 99.9% accuracy. However, our models did not generalize as good on the critical data, attaining only an accuracy of 76.5% as opposed to 95.9%. This could be that 400 nodes is not high enough complexity to capture the finer details needed to distinguish the critical states, or it could simply be because the learning rate and penalty was not chosen precise enough: Since the training is very expensive, our grid search is rather course.

Figure 4.13 highlights two states from the critical set that was misclassified by the best neural network trained on the ordered/disordered set.

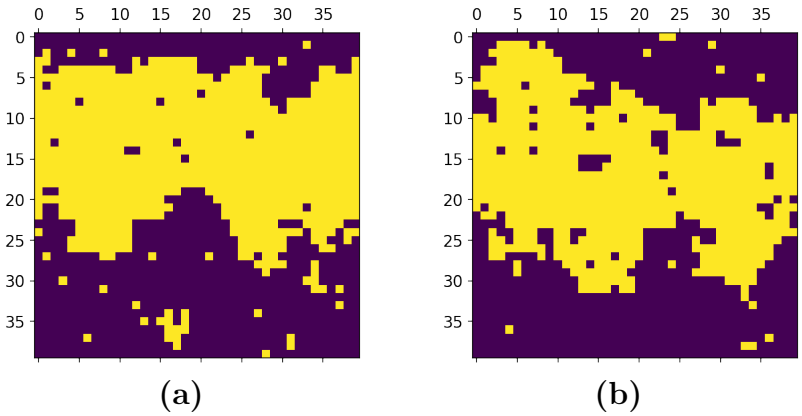



**Figure 4.13:** *Examaple of two states from the critical set misclassified by the best network trained on the ordered/disordered set.*

Purely visually, one can see where the ambiguity in classification occurs: The misclassified states looks locally ordered in some places and disordered in other. This ambiguity can be theorized to confuse the neural network and produce misclassification.

# 5. Conclusion

In this project, we have applied several statistical learning methods to data from both the one-dimensional and two-dimensional Ising model. In the one-dimensional case, the linear regression methods OLS, Ridge and Lasso were used to determine the coupling constant for the Ising model Hamiltonian. A multilayer neural network was also applied on the regression problem. The two-dimensional Ising model, on the other hand, exhibits a phase transition from a magnetic phase to a phase with zero magnetization at a given critical temperature, called the Curie temperature $T_C$. Both logistic regression and neural networks was used in order to classify the phase of the two-dimensional Ising model.

In the linear regression problem, we assumed no prior knowledge about the origin of the data set. Hence, a Ising model with pairwise interactions between every pair of variables was used. However, this generalized model gives rise to that OLS and Ridge regression learn nearly symmetric weights $J = -0.5$. Lasso regression, however, tends to break this symmetry and perfectly determine the coupling $J = -1$ with proper regularization. With regularization $\lambda = 10^{-2}$, Lasso gave an $R^2$ score of 1.

The two-dimensional Ising model exhibits a phase transition from an ordered phase to a disordered phase at the Curie temperature. The goal was to build a model which will take in a spin configuration and predict whether it constitutes an ordered or disordered phase. The provided dataset consisted of ordered, disordered and "critical-like" phases. Both ordered and disordered states were used to train the logistic regressor. The remaining critical states was used as a held-out validation set which the hope was that we would be able to make good extrapolated predictions on. If successful, this may have been a viable method to locate the position of the critical point in other complicated models with no known exact analytical solution. However, logistic regression was not able to correctly fit the Ising model as it is not a linear model, and was, in the case with both Gradient Descent and Newton-Raphson as optimization methods, no better than just guessing the phase.

Neural networks proved useful to predict the energy of the Ising model (Figure 4.9), even in the presence of a high number of useless features produced by the generalized Ising model. The final R2-score on test data was 94.3%. Regularization was proved crucial in order to eliminate the influence of features that did not contribute to the energy (Figure 4.10). Compared to OLS and Ridge for determining coupling constants, which is a similar problem, neural networks performed generally better, though not as good as Lasso.

We managed to reproduce the same order of accuracy when determining the phase of the Ising model as Mehta, for neural networks trained on the ordered/disordered set(Figure 4.12). However, our models did not generalize as good on the critical data set. This may be a shortcoming of our simpler network, using only 400 nodes instead of 1000 as Mehta. This was chosen because of computationally limitations. Another major factor is our very coarse grid search, testing for only 3 learning rates and penalties.

# 6. Future Work

The `LogisticRegression` class should be extended to include other optimization methods, such as Stochastic Gradient Descent and Mini-batch Gradient Descent. As became evident from our logistic regression analysis, standard Gradient Descent and Newton-Raphson's method are slow to use on large training instances. Both Stochastic Gradient Descent and Mini-Batch Gradient Descent should perform better on these cases.

Implementing random search rather than grid search when optimizing hyper parameters will likely result in better models, since grid search is very computationally heavy. Further, doing the same neural network analysis using frameworks such as Tensorflow, Keras or Pytouch would provide a helpfull benchmark for comparison.

# References


[1] Pankaj Mehta et al. "A high-bias, low-variance introduction to Machine Learning for physicists". In: *Physics Reports* 810 (May 2019), pp. 1–124. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2019.03.001. URL: http://dx.doi.org/10.1016/j.physrep.2019.03.001 (cit. on pp. 1, 9, 11, 21).

[2] N. Haug. *FYS3150 Project 4 - Studies of Phase Transitions in Magnetic Systems Using the Square Lattice Ising Model and the Metropolis Algorithm*. URL: https://github.com/nicolossus/FYS3150/blob/master/Project4/Report/FYS3150_Project4.pdf (cit. on p. 2).

[3] Lars Onsager. "Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition". In: *Phys. Rev.* 65 (3-4 Feb. 1944), pp. 117–149. DOI: 10.1103/PhysRev.65.117. URL: https://link.aps.org/doi/10.1103/PhysRev.65.117 (cit. on p. 2).

[4] N. Haug, T. K. Netskar, and K. Wold. *FYS-STK4155 Project 1 - Regression Analysis and Resampling Methods*. URL: https://github.com/nicolossus/FYS-STK4155/blob/master/Project1/FYS_STK4155_Project_1.pdf (cit. on pp. 2, 9, 26).

[5] M. H Jensen. *Data Analysis and Machine Learning: Logistic Regression*. URL: https://compphysics.github.io/MachineLearning/doc/pub/LogReg/html/LogReg.html (cit. on pp. 3–5).

[6] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. eng. 1st ed. O'Reilly Media, Inc, 2017. ISBN: 9781491962299 (cit. on pp. 3–5, 29).

[7] M. H. Jensen. *Predicting phase transition of the two-dimensional Ising model*. URL: https://github.com/CompPhysics/MachineLearning/blob/master/doc/Programs/IsingModel/logisticIsing.ipynb (cit. on p. 5).

[8] Tom Lindstrøm and Klara Hveberg. *Flervariabel analyse med lineær algebra*. nor. New York, NY: Pearson Education Limited, 2011. ISBN: 9780273738138 (cit. on p. 6).

[9] M. H. Jensen. *Data Analysis and Machine Learning:Neural networks, from the simpleperceptron to deep learning*. URL: https://compphysics.github.io/MachineLearning/doc/pub/NeuralNet/pdf/NeuralNet-minted.pdf (cit. on p. 7).

[10] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. eng. Second Edition. Springer Series in Statistics. New York, NY: Springer New York, 2009. ISBN: 9780387848570 (cit. on pp. 26, 28).

# A. Review of Linear Regression

The following material is an excerpt from the theory section in [4].

## A.1. Regression Analysis

Let $X = (X_1, \ldots, X_p) \in \mathbb{R}^p$ and $Y \in \mathbb{R}$ be random variables. A regression model assumes that there is a function $f\colon \mathbb{R}^p \to \mathbb{R}$ such that

$$\mathrm{E}(Y|X) = f(X).$$

The components of $X$ are referred to as the *inputs*, *predictors* or *independent variables*, while $Y$ is referred to as the *output*, *response* or *dependent variable* [10, p. 9].

We will use the notation $\hat{f}$ for a function that is an estimator of $f$. A value $\hat{Y} = \hat{f}(X)$ is then referred to as the *fitted* or *predicted value* at input $X$. The estimator $\hat{f}$ is usually calculated from a collection $\{(X_i, Y_i)\, i = 1, \ldots, N\}$ of observations. For this reason this collection is often called a training set.

The *bias* of an estimator $\hat{f}$ in an input point $X$ is the difference

$$\mathrm{Bias}(\hat{f}(X)) = \mathrm{E}(\hat{f}(X)|X) - f(X)$$

between the expected estimate and the true value in $X$. If the bias is 0, we say that the estimator is *unbiased*. Otherwise the estimator is *biased*.

A linear regression model assumes that $f$ is such that

$$f(X) = \beta_0 + \sum_{i=1}^{p} X_i \beta_i$$

for some $\beta = (\beta_0, \ldots, \beta_p)^T \in \mathbb{R}^{p+1}$. It is customary to include 1 as the first component in $X$, so that $X = (1, X_1, \ldots, X_p)^T \in \mathbb{R}^{p+1}$ and one can write

$$\mathrm{E}(Y|X) = X^T\beta.$$

## A.2. Ordinary Least Squares (OLS)

Assume a linear regression model $\mathrm{E}(Y|X) = X^T\beta$, $\beta \in \mathbb{R}^{p+1}$, for the random variables $X = (1, X_1, \ldots, X_p)^T \in \mathbb{R}^{p+1}$ and $Y \in \mathbb{R}$. The ordinary least squares is a method for estimating $\beta$. Namely, given a training set $\{(x_i, y_i) \in \mathbb{R}^{p+1} \times \mathbb{R} : i = 1, \ldots, N\}$, we choose the coefficients $\beta$ that minimize the residual sum of squares

$$\mathrm{RSS}(\beta) = \sum_{i=1}^{N} \left(y_i - x_i^T\beta\right)^2.$$

Since this is a quadratic function in the parameters $\beta$, it always has a minimum, but it need not be unique. By writing $\mathbf{X}$ for the $N \times p$-matrix with $x_1, \ldots, x_n$ as rows and $\mathbf{y}$ for the column vector with components $y_1, \ldots, y_n$, we get the residual sum of squares in matrix form as

$$\mathrm{RSS}(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta).$$

By the chain rule and the fact that the Jacobian matrix of $a \mapsto a^T a$ is $2a^T$, we get that the gradient of RSS is

$$\nabla\mathrm{RSS}(\beta) = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta).$$

The Hesse matrix $H\text{RSS}(\beta)$ is equal to the Jacobian matrix of $\beta \mapsto \nabla\text{RSS}(\beta)$, thus

$$H\text{RSS}(\beta) = \mathbf{X}^T\mathbf{X}.$$

Assume now that $\mathbf{X}$ has full column rank. Then $\mathbf{X}^T\mathbf{X}$ is positive definite and in particular invertible. It follows that the equation

$$\nabla\text{RSS}(\beta) = 0 \quad \Leftrightarrow \quad \mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) = 0$$

has the unique solution

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}, \tag{A.1}$$

which gives the unique minimum of RSS. The predicted value at an input vector $x_0$ (with 1 as the first component) is $\hat{f}(x_0) = x_0^T\hat{\beta}$. The fitted values at the training inputs are

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Note that

$$\text{RSS}(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) = \|\mathbf{y} - \mathbf{X}\beta\|^2,$$

so a vector $\hat{\beta}$ minimizing RSS is such that $\mathbf{X}\hat{\beta}$ is the point in the column space of $\mathbf{X}$ that is closest to $\mathbf{y}$. This is valid also in the case when RSS does not have a unique minimum. In other words, $\mathbf{X}\hat{\beta}$ is the orthogonal projection of $\mathbf{y}$ onto the column space of $\mathbf{X}$. The matrix

$$\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$$

is the projection matrix onto the column space of $\mathbf{X}$ (with the property $\mathbf{H}^T = \mathbf{H} = \mathbf{H}^2$). It is often called the "hat" matrix, since it puts a hat on $\mathbf{y}$.

Using linearity of the expected value we find that

$$\text{E}(\hat{\beta}|\mathbf{X}) = \text{E}((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}|\mathbf{X}) = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\text{E}(\mathbf{y}|\mathbf{X}) = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}\beta = \beta.$$

Moreover, since in general $\text{Var}(\mathbf{A}Z) = \mathbf{A}\text{Var}(Z)\mathbf{A}^T$ for constant matrices $\mathbf{A}$ and random vectors $Z$ (by the linearity of the expectation), it follows that

$$\text{Var}(\hat{\beta}|\mathbf{X}) = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\text{Var}(\mathbf{y})\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\sigma^2 I\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}. \tag{A.2}$$

A linear transformation of a multivariate normal random vector is again multivariate normal. Thus $\hat{\beta} \sim N(\beta, \sigma^2(\mathbf{X}^T\mathbf{X})^{-1})$.

We usually estimate the variance $\sigma^2$ by

$$\hat{\sigma}^2 = \frac{1}{N-p-1}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2.$$

Plugging this into Equation (A.2), we get an estimate of $\text{Var}(\hat{\beta})$ by

$$\hat{\sigma}^2(\mathbf{X}^T\mathbf{X})^{-1} = \frac{1}{N-p-1}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2(\mathbf{X}^T\mathbf{X})^{-1}. \tag{A.3}$$

It can be shown that $\hat{\sigma}$ has a distribution such that

$$(N-p-1)\hat{\sigma}^2 \sim \sigma^2\chi^2_{N-p-1},$$

the chi-squared distribution with $N-p-1$ degrees of freedom.[10, p. 47] Moreover, the random variables $\hat{\beta}$ and $\hat{\sigma}^2$ are statistically independent.

By writing the parameters $\hat{\beta}_j$ as linear transformations of $\hat{\beta}$, we can show that $\hat{\beta}_j \sim N(\beta_i, \sigma^2 v_j)$, where $v_j$ is the $j$-th diagonal element of $(\mathbf{X}^T\mathbf{X})^{-1}$, or, in other words $\sigma^2 v_j$ is the $j$-th diagonal element of $\mathrm{Var}(\hat{\beta})$.

Under the null hypothesis that $\beta_j = 0$, the random variable

$$z_j = \frac{\hat{\beta}_j}{\hat{\sigma}\sqrt{v_j}}$$

is distributed as $t_{N-p-1}$, a $t$ distribution with $N-p-1$ degrees of freedom.

## A.3. Ridge Regression

Ridge regression is an example of a method that gives a biased estimator. It is similar to least squares, but instead of minimizing the sum $\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$, we minimize

$$\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{i=1}^{p} \beta_i^2, \tag{A.4}$$

where $\lambda$ is some positive real number. The consequence, compared to least squares, is that the coefficients $\beta$ are shrunken. The amount of shrinkage is controlled by the value of $\lambda$. Larger $\lambda$ means more shrinkage. The solution is denoted $\hat{\beta}^{\text{ridge}}$.

It can be shown that an equivalent way of formulating the ridge problem is as

$$\hat{\beta}^{\text{ridge}} = \arg\min_{\beta} \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$$

subject to the constraint

$$\sum_{i=1}^{p} \beta_i^2 \leq t.$$

for some $t > 0$. There is a one-to-one correspondence between the parameters $\lambda$ and $t$. It can be shown [10, Exercise 3.5] that if we replace each $x_{ij}$ by $x_{ij} - \overline{x}_j = x_{ij} - \frac{1}{N}\sum_{i=1}^{N} x_{ij}$, the ridge problem has the following two step solution: First estimate $\beta_0$ by $\overline{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$, then estimate the remaining coefficients by a ridge regression without the intercept. If we remove the intercept from $\beta$ and remove the column of ones from the design matrix $\mathbf{X}$, we can write the expression to minimize in the ridge regression step as

$$(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\beta^T\beta = \|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda\|\beta\|^2$$

The ridge regression solution can then be shown to be

$$\hat{\beta}^{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{y}. \tag{A.5}$$

Adding a strictly positive constant $\lambda I$ to $\mathbf{X}^T\mathbf{X}$ makes the problem nonsingular also in the case where $\mathbf{X}^T\mathbf{X}$ is not invertible.

Similarly to for OLS, we can show that

$$\mathrm{E}(\hat{\beta}^{\text{ridge}}|\mathbf{X}) = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{X}\beta$$

and

$$\mathrm{Var}(\hat{\beta}^{\text{ridge}}|\mathbf{X}) = \sigma^2(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}. \tag{A.6}$$

This shows that the ridge estimator is unbiased if and only if $\lambda = 0$.

## A.4. Lasso Regression

Lasso (least absolute shrinkage and selection operator) is a regression method where we choose the coefficients $\beta$ by minimizing the sum

$$\frac{1}{2}\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{i=1}^{p}|\beta_i|, \tag{A.7}$$

for some $\lambda > 0$. The solution is denoted $\hat{\beta}^{\text{lasso}}$. Equivalently,

$$\hat{\beta}^{\text{lasso}} = \arg\min_{\beta}\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$$

subject to the constraint

$$\sum_{j=1}^{p}|\beta_j| \leq t$$

for some $t > 0$. The parameters $t$ are in a one-to-one correspondence with the parameters $\lambda$.

## A.5. Summary Statistics

Training a model means setting its parameters so that the model best fits the training set. In order to find the best fit, we need to measure how well the model performs on the training data. A common performance measure is the Mean Square Error (MSE)[6, p. 109]:

$$\text{MSE}\left(\hat{y}, \tilde{\hat{y}}\right) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2, \tag{A.8}$$

where $\tilde{y}_i$ is the predicted value of the $i$th sample and $y_i$ is the corresponding true value. In order to train a regression model, we need to find the value of $\beta$ that minimizes the MSE.

The $R^2$ score function is also a useful statistic in the analysis of how well a model performs. It is a measure of how close the data are to the fitted regression line, and is given by

$$R^2\left(\hat{y}, \tilde{\hat{y}}\right) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \tilde{y})^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2} = 1 - \frac{\text{MSE}}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2}, \tag{A.9}$$

where the mean value of $y_i$ is defined as

$$\bar{y} = \frac{1}{n}\sum_{i=0}^{n-1} y_i$$

## A.6. Bias-variance Tradeoff

Suppose that we have a regression function $f$ such that

$$Y = f(X) + \varepsilon$$

for random variables $X, Y, \varepsilon \in \mathbb{R}$, where $\varepsilon \sim N(0, \sigma^2)$. Let $\hat{f}$ be an estimator for $f$ and let $(X, Y) = (X, f(X) + \varepsilon)$ be an observation independent from $\hat{f}$. For $X = x_0$ we get that the

expected prediction error is

$$\begin{aligned}\mathrm{E}((Y-\hat{f}(x_o))^2) &= \mathrm{E}[((Y-f(x_0))+(f(x_0)-\mathrm{E}(\hat{f}(x_0)))+(\mathrm{E}(\hat{f}(x_0))-\hat{f}(x_0)))^2] \\ &= \mathrm{E}[(Y-f(x_0))^2+(f(x_0)-\mathrm{E}(\hat{f}(x_0)))^2+(\mathrm{E}(\hat{f}(x_0))-\hat{f}(x_0))^2 \\ &+ 2(Y-f(x_0))(f(x_0)-\mathrm{E}(\hat{f}(x_0)))+2(Y-f(x_0))(\mathrm{E}(\hat{f}(x_0))-\hat{f}(x_0)) \\ &+ 2(f(x_0)-\mathrm{E}(\hat{f}(x_0)))(\mathrm{E}(\hat{f}(x_0))-\hat{f}(x_0))].\end{aligned}$$

Since $Y$ is independent from $\hat{f}(x_0)$ and $\mathrm{E}(Y)=f(x_0)$, it follows that

$$\begin{aligned}\mathrm{E}((Y-\hat{f}(x_o))^2) &= \mathrm{E}((Y-f(x_0))^2)+(f(x_0)-\mathrm{E}(\hat{f}(x_0)))^2+\mathrm{E}((\mathrm{E}(\hat{f}(x_0))-\hat{f}(x_0))^2) \\ &= \sigma^2+\mathrm{Bias}^2(\hat{f}(x_0))+\mathrm{Var}(\hat{f}(x_0)).\end{aligned}$$

This is referred to as the bias-variance decomposition of the expected prediction error. The first term is the variance of $f(x_0)$. The second is the squared bias of $\hat{f}(x_0)$, and measures how far $\hat{f}(x_0)$ is from the true value $f(x_0)$. The last term is the variance of the estimator in the point $x_0$.

The bias-variance decomposition gives a motivation to consider biased estimators. To get the lowest expected test error, we must find the right balance between the bias and the variance of the estimator. If the variance is sufficiently low, it is worthwhile to have a little bias.