
# UNIVERSITÀ DEGLI STUDI DI PADOVA

**Dipartimento di Fisica e Astronomia "Galileo Galilei"**

**Corso di Laurea in Fisica**


**Tesi di Laurea Magistrale**

# Machine learning techniques applied to the statistical properties of spin models


**Relatore**

**Prof. Enzo Orlandini**

**Correlatore**

**Prof. Marco Baiesi**

**Laureando**

**Matteo Ferraretto**


**Anno Accademico 2018/2019**

# Contents

# Introduction

Machine learning (ML) is a general term used to identify a set of algorithms and computing techniques that, although theorized in the last decades of the last century, have been widely developed in the last twenty years thanks both to technological improving of hardwares, and to optimization of algorithms. The goal of ML is to detect deep abstract features and patterns from data, in order to develop a model capable of predicting features of new, unobserved data. This goal is achieved through a set of hidden parameters which values have to be determined from data. ML techniques are divided into two main cathegories depending on the available dataset: supervised machine learning (SML) and unsupervised machine learning (UML). The former is used when labeled input data are available; they are used to set the values of the hidden parameters, hence to label new unseen data according to the abstract detected pattern. The latter is used when dealing with unlabeled data from which one wishes to extract features.

Several applications of Machine Learning have been investigated recently through research and technological contexts. ML have been successfully applied to image recognition [14], image segmentation [15], human speech recognition and elaboration [16, 17], grammatical processing of natural speaking [18], recognition of semantic patterns and authomatic translation [19], subatomic particle detection [20], diagnosis and prognosis problems in medicine [21], and many more.

Very recently, the physicists community has begun to study machine learning on two sides: both applying it to detect features of physical problems, and applying physical tools to understand deep properties of the algorithms. The focus of this work is on the former research line, and in particular we investigate some physical problems that can be addressed with machine learning. Starting from a physical question, we present the main techniques that can be used to answer, then we apply them to simple systems with well known properties, such as Ising, Potts and XY models. For every question, we not only provide a summary of the state of the art but we also implement all the algorithms ourselves, using different languages for different tasks, in particular C++, Wolfram Mathematica 12.0 and Python. The most relevant code lines for each topic are collected in Appendix, so that the interested reader is encouraged to reproduce and eventually improve our results.

The thesis is organized as follows. Chapter 1 summarizes the main properties of the physical models that we have used as a benchmark throughout the work, and it gives an overview of Monte Carlo sampling, explaining how we have built up the starting datasets. Chapter 2 deals with the problem of designing the simplest physical model compatible with a given set of configurations and the relative energies. We show that such problem can be easily solved by determining the coupling constants of the model with a linear regression. The problem though is more subtle than a simple linear fit: since there are many parameters to determine, and since we want a model which is able to predict energies of unseen configurations in the phase space, we encounter the problem of overfitting. In order to avoid overfitting we introduce regularized regressions, that are the simplest examples of machine learning tasks, so we can introduce the first key concepts of ML.

Chapter 3 and 4 discuss the physical problem of predicting the phase of a given configuration (ordered or disordered) in a system undergoing phase transition. More generally we face the problem of labeling data. In Chapter 3 we use a labeled dataset to build a classifier that can label a new unseen configuration. The basic tools to face this problem are neural networks. In this chapter we discuss in detail some basic neural network architectures (feed forward and convolutional), we study how they

learn to classify and we compare their efficiencies. The starting problem here is just a basic example to get familiar with these tools, but in principle they can be extended to more ambitious tasks (predicting the most likely sampling temperature of a given configuration, predicting energy, generating large systems etc.) and to other fields of physics. In Chapter 4 we perform data labeling without having previously labeled data available (unsupervised learning). The task can be accomplished with a dimensional reduction of the input, namely representing each $n$-dimensional data-point (described through $n$ features) with a much smaller number of most relevant features. Another basic tool of UML analyzed in Chapter 4 is clustering, a process that allows classification of data into clusters made by neighbouring points in phase space.

# Chapter 1

# Overview of spin models and Monte Carlo sampling

This chapter is organized as follows. The first part is dedicated to a brief overview of some of the most relevant spin models studied in Statistical Physics. In particular we will focus on the Ising, Potts and XY models, which are object of study in the following chapters. A review of the main theoretical results about symmetry properties and critical behaviour of the models is provided. The last part is dedicated to a general introduction to Monte Carlo tecniques for sampling equilibrium configurations of the models, with a detailed description of the Metropolis-Hastings algorithm, that we have implemented to extract samples for further analysis.

## 1.1 Ising model

The Ising model was originally introduced and solved by Ernst Ising in 1924 for modeling the paramagnetic to ferromagnetic phase transition in some materials. It has achieved resounding success not only because it can be solved (i.e. the partition function and associated thermodynamical properties can be found exactly in simple geometries), but because many physical models have the same critical properties and belong to the same universality class. Moreover, since its properties are well-known and relatively simple, this model can be used as a test-bed for any theoretical and computational tecniques.

The dynamical variables of the model are called spins, and they can be either $+1$ or $-1$. They form a discrete set which is organized in a $d$-dimensional lattice where a notion of distance is introduced. The Ising Hamiltonian is

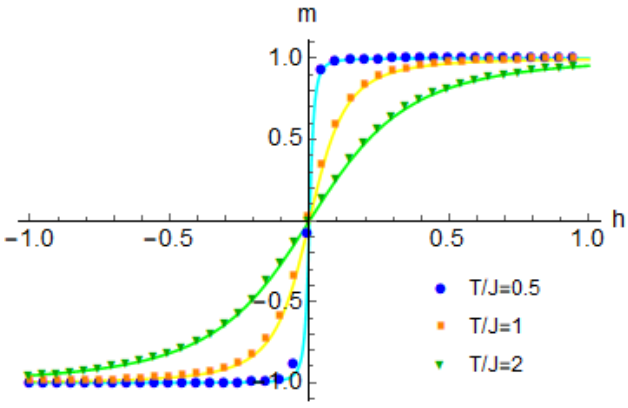
$$\mathcal{H} = -J \sum_{\langle ij \rangle} S_i S_j - H \sum_i S_i \tag{1.1}$$

where $J$ is the coupling constant between spins and $H$ is an external uniform field. The notation $\langle ij \rangle$ means that $i$ and $j$ are nearest neighbors sites of the lattice. When the system is not coupled to an external field ($H = 0$), then the Hamiltonian ($\mathcal{H}_0$) is invariant under flipping of all spins. In other words the free Hamiltonian has a global $Z_2$ symmetry.

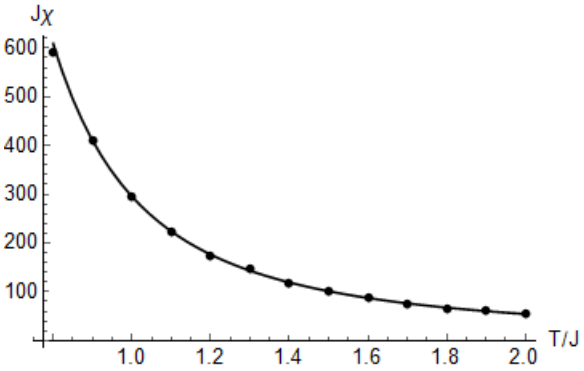
The partition function is the sum of the canonical Boltzmann probabilities on all the possible configurations

$$\mathcal{Z}(J, T, H) = \sum_{\{S\}} e^{-\beta \mathcal{H}} \equiv \mathrm{Tr} \left[ e^{-\beta \mathcal{H}} \right] \tag{1.2}$$

where $\beta = 1/k_B T$ and from now on we will use the convention $k_B = 1$. The ensemble average of any observable $A$ can be computed using the partition function, namely $\langle A \rangle = \mathcal{Z}^{-1} \mathrm{Tr} \left[ A e^{-\beta \mathcal{H}} \right]$. An interesting physical observable is the sum of all spins, called magnetization ($M$) and its ensemble




(a) Magnetization per site in one dimensional Ising model as a function of the external magnetic field at different temperatures. Dots are data generated with Metropolis-Hastings algorithm with $N = 40$, solid lines are exact predictions.




(b) Magnetic susceptibility obtained by the simulation (dots) and theoretical function (solid line) of the 1-dimensional Ising model.

average is then given by

$$\langle M \rangle = \frac{\mathrm{Tr}\left[M e^{-\beta(\mathcal{H}_0 - HM)}\right]}{\mathcal{Z}(J,T,H)} \tag{1.3}$$

A state with $M = 0$ is called disordered (or paramagnetic phase); a state with $M \neq 0$ is then ordered (or ferromagnetic phase). In the thermodynamic limit (i.e. when the number of lattice sites $N \to \infty$), the magnetization of an ordered state could in principle diverge, so we will often use the magnetization per site $m = M/N$ which is always finite.

Another interesting quantity is the magnetic susceptibility $\chi = \partial M/\partial H|_{H=0}$. This quantity is related to the equilibrium fluctuations of magnetization

$$\chi = \left.\frac{\partial M}{\partial H}\right|_{H=0} = \left.\frac{\partial}{\partial H}\frac{\mathrm{Tr}\left[M e^{\beta\mathcal{H}_0 - HM}\right]}{\mathcal{Z}(T,H)}\right|_{H=0} = \beta\left(\langle M^2\rangle - \langle M\rangle^2\right) \tag{1.4}$$

In terms of the magnetization per site and of the rescaled external field $h = \beta H$ we get

$$\chi = \beta N \left.\frac{\partial m}{\partial h}\right|_{h=0} = \beta N^2\left(\langle m^2\rangle - \langle m\rangle^2\right). \tag{1.5}$$

The expectation value of a product of two spins separated by a distance $\vec{r}$ in the lattice is called correlation function and indicated by $\Gamma(\vec{r})$. In some cases the correlation function exponentially decays as $\Gamma(r) \sim e^{-r/\xi}$. In these cases $\xi$ is called correlation length and in principle it depends on $J$, $T$ and $H$. This function is useful when studying the critical behaviour of the system, not only in the Ising case.

Let's now consider a one dimensional lattice with periodic boundary conditions ($S_{N+1} = S_1$). In this simple geometry, the partition function can be exactly computed using the transfer matrix method. The resulting magnetization per site is

$$m(K,h) = \frac{\sinh h + \frac{\sinh h \cosh h}{\sqrt{\sinh h^2 + e^{-4K}}}}{\cosh h + \sqrt{\sinh h^2 + e^{-4K}}} \tag{1.6}$$

where $K = \beta J$. From this result it is clear that when $h = 0$, the magnetization per site vanishes at all temperatures. In other words, when the one dimensional system has a global $Z_2$ symmetry, the ground state is always disordered and it is invariant under the spin flipping as well. In the language of group theory this means that there is no spontaneous symmetry breaking of $Z_2$ and no phase transition between paramagnet and ferromagnet occurs. In Fig. 1.1a we show a comparison between

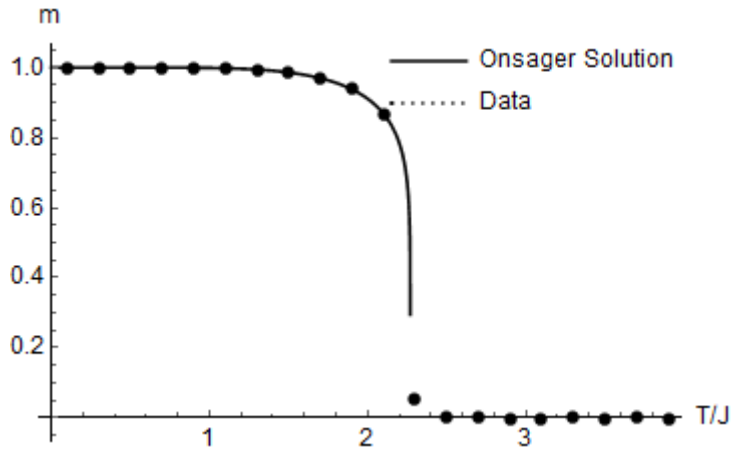



Figure 1.2: Average ensemble magnetization as a function of temperature at zero external magnetic field. The dots are the results obtained by Metropolis-Hastings algorithm with $L = 40$; the solid line is Onsager's exact solution. The phase transition is observed.

theoretical results and the outcome of the Monte Carlo simulation (discussed in the following) for the magnetization per site.

Using Eqs. 1.5 and 1.6 we can find an exact expression for the magnetic susceptibility by deriving the magnetization which respect to $h$. Moreover, we can use the right hand side of Eq. 1.5 to compute the susceptibility by numerical simulations. A comparison between these two results is shown in Fig. 1.1b.

It is also possible to prove that when $h = 0$ the correlation length is given by

$$\xi = -\frac{1}{\log \tanh K} \tag{1.7}$$

for an infinite lattice. Since in the simulations we have to choose a finite size $N$, the correlation length provides a lower limit for the temperature at which we take our samples. In particular we need $\xi \ll N$, otherwise we would see a fictitious ordered phase which is related to the finite size of the system. The temperatures at which we can consistently sample spin configurations is then (after inverting Eq. 1.7)

$$T \gg \frac{2J}{\log \frac{e^{1/N}+1}{e^{1/N}-1}} \tag{1.8}$$

Let's now consider an Ising model in a two dimensional square lattice of size $L$ with $N = L^2$ sites, periodic boundary conditions and no external field. An exact solution for this model in the thermodynamic limit $N \to \infty$ was found by Onsager in 1944 with the transfer matrix method

$$m(K, 0) = \begin{cases} \pm \left[1 - \sinh^{-4} K\right]^{1/8} & \text{if } T < T_c \\ 0 & \text{if } T \geq T_c \end{cases} \tag{1.9}$$

where $T_c$ is the critical temperature $T_c = 2J/\log(1+\sqrt{2}) \approx 2.269J$. As it is clear from Eq. 1.9, this model undergoes a phase transition at $T = T_c$ between a ferromagnetic phase ($T < T_c$) and a paramagnetic phase ($T > T_c$). Since at $T < T_c$ the system is mostly magnetized upwards (or downwards), the ground state does not have the same $Z_2$ symmetry as the Hamiltonian and hence there is a spontaneous symmetry breaking. The magnetization per site is a suitable order parameter for this transition, and since it is a continuous function of $T$, the transition is second order. In Fig. 1.2 we compare Monte Carlo simulations (see next sections) with the Onsager solution of Eq. 1.9.

A key concept in statistical physics is that a system can undergo a phase transition only in the thermodynamic limit. The reason is that a phase transition, by definition, occurs when the partition

function is not analytic at some values of its variables. In a finite size system though, the partition function is a finite sum of exponentials and then it is always analytic. When the system is infinite the sum of infinite terms can in principle diverge and a phase transition is possible. A central problem that arises from this fact is that since we can only perform simulations of finite sized systems, in principle we should never observe any phase transitions. In practice, even though the partition function of a finite system is analytic, we observe an ordered phase when the correlation length is comparable to the lattice size $\xi \sim L$. The correlation length close to the phase transition scales in general as a power law of $T - T_c$ and the critical exponent is indicated by $\nu$

$$\xi \sim (T - T_c)^{-\nu} \tag{1.10}$$

where $\nu = 1$ in the two dimensional Ising model, and $\nu > 0$ in general. If the transition temperature of the finite system $T^*$ is close to the critical value ($T^* - T_c \ll T_c$), then from $\xi \sim L$ we get

$$\frac{T^* - T_c}{T_c} \sim L^{-1/\nu} \quad \rightarrow \quad T^*(L) = T_c + \frac{k}{L^{1/\nu}} \tag{1.11}$$

where $k$ is a constant depending on the details of the model. A consequence of this property is that critical properties of the infinite system can be deduced by extrapolation from the result of numerical simulations at increasing lattice sizes.

## 1.2 Potts model

The Potts model is a generalization of the Ising model which extends the $Z_2$ symmetry of the free Ising Hamiltonian (Eq. 1.1 at $H = 0$) to an Hamiltonian invariant under the action of the group $Z_q$ (group of the permutations of $q \geq 2$ objects). The Hamiltonian of the model is

$$\mathcal{H} = -J \sum_{\langle ij \rangle} \delta_{\sigma_i, \sigma_j} \tag{1.12}$$

where $\sigma = 0, 1, ..., q-1$ is a variable which can assume $q$ different values, $J$ is the nearest neighbors interaction energy. Since the case $q = 2$ can easily be mapped into an Ising model, as we will prove at the end of this section, we start focusing on the $q \geq 3$ case. In particular we present a mean field approach to the solution.

Let $x_\sigma$ be the fraction of sites in the state $\sigma$, the probability that a lattice site be in a given state $\bar{\sigma}$ can be written as

$$p(\sigma) = \sum_\rho x_\rho \delta_{\rho,\sigma} = x_\sigma \tag{1.13}$$

We now have to write down the average free energy $F = \langle \mathcal{H} \rangle - T \langle S \rangle$ of the system using this probability. The average entropy is the sum of the entropies associated to every lattice site $-\sum_\sigma x_\sigma \log x_\sigma$, so that

$$\langle S \rangle = -N \sum_\sigma x_\sigma \log x_\sigma \tag{1.14}$$

The average internal energy is easy in the mean field approximation for $\left\langle \delta_{\sigma_i,\sigma_j} \right\rangle \approx \sum_\sigma \langle \delta_{\sigma_i,\sigma} \rangle \left\langle \delta_{\sigma_j,\sigma} \right\rangle = \sum_\sigma \langle \delta_{\sigma_i,\sigma} \rangle^2$. The average can be computed using the probability distribution introduced above

$$\langle \delta_{\sigma_i,\sigma} \rangle = \sum_{\sigma_i} p(\sigma_i) \delta_{\sigma_i,\sigma} = p(\sigma) = x_\sigma$$

In mean field this quantity is assumed to be independent on the lattice size, so that

$$\langle \mathcal{H} \rangle = -J \sum_{\langle ij \rangle} \sum_\sigma x_\sigma^2 = -\frac{JNz}{2} \sum_\sigma x_\sigma^2, \tag{1.15}$$

where $z$ is the coordination number of the lattice and the factor 2 avoids double counting. At this point we can make an educated guess on $x_\sigma$

$$x_0 = \tfrac{1+(q-1)s}{q} \qquad x_\sigma = \tfrac{1-s}{q} \tag{1.16}$$

where $s \in [0, 1[$ is an order parameter, which takes into account the $Z_q$ spontaneous symmetry breaking: if $s = 0$ the system is disordered, since $x_\sigma = 1/q \ \forall\sigma$; while if $s \neq 0$ the majority of the lattice sites are in the state $\sigma = 0$, and the symmetry is broken. Moreover the consistency constraint $\sum_\sigma x_\sigma = 1$ is satisfied.

After some calculations one can finally write the free energy per site $f = F/N$ as a function of $s$ (neglecting an additive constant):

$$\beta f(s) = \frac{1+(q-1)s}{q}\log\left[1+(q-1)s\right] + \frac{(q-1)(1-s)}{q}\log\left(1-s\right) - \frac{q-1}{2q}zKs^2 \tag{1.17}$$

where $K = \beta J$ as previously defined. The order parameter as a function of the temperature is obtained solving $f'(s) = 0$, since it is the minimum of the free energy. This leads to the implicit equation

$$\log\frac{1+(q-1)s}{1-s} = zKs \tag{1.18}$$

This equation has only the trivial solution $s = 0$ for small values of $zK$ (high temperature), three solutions $s = 0$, $s = s_1(zK)$ and $s = s_2(zK)$ (with $s_1 < s_2$) in an intermediate range of $zK$, and two solutions $s = 0$ and $s = s_2(zK)$ for large values of $zK$ (low temperature). One can also check that $s_2$ is always a local minimum of the function, but not necessarily a global minimum.

The global minimum of $f(s)$ is $s = 0$ for small values of $zK$, and becomes $s = s_2(zK)$ when increasing $zK$ above a critical value $zK_c$, meaning that the system undergoes a phase transition. Since $s_2(zK_c) \equiv s_c \neq 0$, the global minimum changes abruptly from $s = 0$ to $s = s_c$: the phase transition is then first order (see Fig. 1.3).

Since $f(0) = 0$, the critical values $K_c$ and $s_c$ are found by solving $f(s) = 0$ and $f'(s) = 0$ simultaneously, and the solutions are

$$zK_c = 2\tfrac{q-1}{q-2}\log\left(q-1\right) \qquad s_c = \tfrac{q-2}{q-1}. \tag{1.19}$$

The critical temperature in units of $zJ$ is then

$$k_B T_c = \frac{q-2}{2(q-1)\log\left(q-1\right)}zJ. \tag{1.20}$$

In particular, for a two dimensional square lattice $z = 4$ and in the limit of large $q$, one obtains $k_b T_c \approx 2J/\log q$.

It is important to remark that the mean field theory is a good approximation for high dimensional systems ($d \geq 3$), or for systems with large $q$ and $d = 2$; while it fails for $d = 1$ (see [10] for details). However in a $d = 2$ square lattice, an exact solution exists for every $q$ and is proved to be

$$k_B T_c = \frac{1}{\log\left(1+\sqrt{q}\right)}J. \tag{1.21}$$

In the limit $q \gg 1$ this becomes $k_B T_c \approx 2J/\log q$, as predicted in mean field approximation. Moreover, the transition is continuous for $q < 4$ and first order for $q \geq 4$ (in higher dimensions the critical value of $q$ marking the border between first order and continuous transition decreases). Details on the proof and extensions to other geometries and dimensions is provided in [10, 12].

As a final remark, we show that the $q = 2$ Potts model can be mapped into an Ising model. Let $\sigma_i = \pm 1$ be the feature at site $i$ in a two-state Potts model. Taking the following identity into account

$$\delta_{\sigma_i,\sigma_j} = \frac{1+\sigma_i\sigma_j}{2}, \tag{1.22}$$

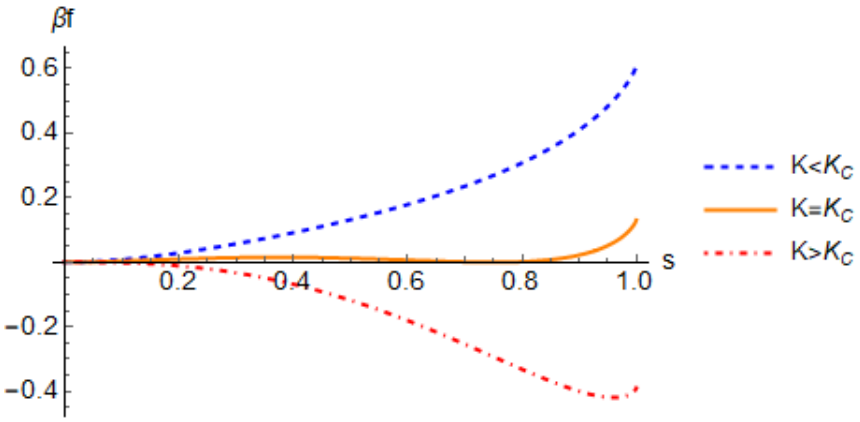



Figure 1.3: [Color online]. Plot of the free energy of the Potts model for $q = 5$ at three different temperatures: $T > T_c$ for the blue dashed line, $T = T_c$ for the orange solid line and $T < T_c$ for the dot-dashed red line.

one can rewrite the Potts Hamiltonian as follows

$$\mathcal{H} = -J_{Potts} \sum_{\langle ij \rangle} \delta_{\sigma_i,\sigma_j} = -J_{Potts} \sum_{\langle ij \rangle} \frac{1 + \sigma_i \sigma_j}{2} = \mathcal{H}_0 - J_{Ising} \sum_{\langle ij \rangle} \sigma_i \sigma_j, \tag{1.23}$$

where $\mathcal{H}_0$ is an irrelevant constant term and $J_I = J_p/2$. This is clearly the Ising Hamiltonian 1.1 and we have found the correct mapping between the models. From Eq. 1.21 we can thus recover Onsager's critical temperature:

$$k_B T_c = \frac{1}{\log(1+\sqrt{2})} J_{Potts} = \frac{2}{\log(1+\sqrt{2})} J_{Ising}.$$

## 1.3 XY model

The $XY$ model is a generalization of the Ising model where the spins are considered as two dimensional vectors and hence the dynamical variables are a set of real numbers representing the angles between a spin and the $\hat{x}$ axis. The Hamiltonian is

$$\mathcal{H} = -J \sum_{\langle ij \rangle} \vec{S}_i \cdot \vec{S}_j = -J \sum_{\langle ij \rangle} \cos(\theta_i - \theta_j) \tag{1.24}$$

The symmetry group of this Hamiltonian is $O(2)$, which is a continuous Lie group and this is the main difference which respect to the Ising and Potts models, whose symmetry groups are discrete ($Z_q$). The Mermin-Wagner theorem states that a spontaneous symmetry breaking of a continuous symmetry group is not possible in systems with dimensionality $d \leq 2$. Since we will focus on the XY model in a two dimensional square lattice, this theorem states that the ferromagnetic phase with non-vanishing magnetization is not possible in our system.

The interesting feature of this model is that it shows signs of critical behaviour, even if a symmetry breaking is not possible. This phase transition has been investigated and explained in detail by Kosterlitz and Thouless and was worth the Nobel prize in Physics in 2016. Here we present only the key features of this transition without going into details of calculations.

The first important aspect concerns the spin correlation function

$$\Gamma(\vec{r}) = \left\langle \vec{S}_0 \cdot \vec{S}_{\vec{r}} \right\rangle = \langle \cos(\theta_0 - \theta_{\vec{r}}) \rangle \tag{1.25}$$

where $\vec{r}$ represents the vector position of a site which respect to the origin. Let $a$ be the lattice spacing, then it turns out that the long distance behaviour of $\Gamma(r)$ strongly depends on temperature, in particular

$$\Gamma(r) \approx_{r \gg a} \begin{cases} \left(\frac{a}{r}\right)^{k_B T/2\pi J} & T \ll T_{KT} \\ e^{-\frac{r}{a} \log \frac{k_B T}{J}} & T \gg T_{KT} \end{cases} \tag{1.26}$$

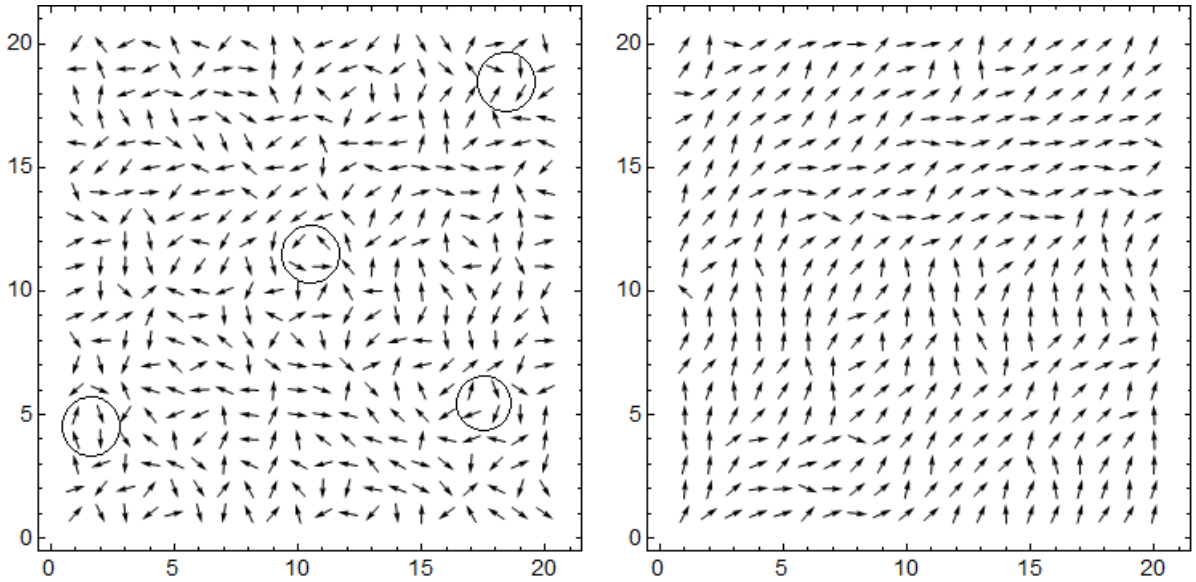



Figure 1.4: Equilibrium configurations of the XY model in a two dimensional square lattice of size $L = 20$. A high-temperature configuration at $k_BT = 1.6J$ is shown in the left panel, the disordered structure with uncoupled vortices and antivortices of small size is presented. A low-temperature configuration at $k_BT = 0.45J$ is shown in the right panel: the quasi-ordered structure with no vortices of small size is shown.

Eq. 1.26 shows that, as expected, the high temperature behaviour of spin correlations decays exponentially with a characteristic length $\xi(T) = a/\log(k_BT/J)$ which is smaller when increasing temperature. In the opposite limit of low temperatures, the spin correlation is much stronger as it decays with a power law with a temperature dependent exponent $\eta(T) = k_BT/2\pi J$. Since a power law decay of correlations is known to be a marker of critical behaviour, we deduce that a phase transition with no spontaneous symmetry breaking occurs at $k_BT = k_BT_{KT} \approx 0.89J$.

The nature of such a phase transition is topological, since it concerns topological structures of spins called vortices, which are typical of two dimensional systems. Approximating the discrete set of real variables $\theta_i$ with a continuous vector field $\theta(\vec{r})$ we can provide a good definition of vortex based on the result of integration over a closed loop of $\nabla\theta$. Given a closed curve $\mathcal{C}$ with winding number $\gamma_{\mathcal{C}} = 1$ in the two dimensional space, the following identity holds

$$\oint_{\mathcal{C}} \nabla\theta \cdot d\vec{l} = 2\pi q \tag{1.27}$$

where $q$ is an integer. This is a consequence of the fact that $\theta = 0, 2\pi, 4\pi$ etc. represent the same physical angle. We say by definition that a closed curve $\mathcal{C}$ contains a vortex of charge $q$, and if $q < 0$ we call them antivortices. The basic idea of Kosterlitz and Thouless is that at low temperatures a vortex and an antivortex are bound together, namely it is possible to choose a path $\mathcal{C}$ surrounding a relatively small region such that $q = 0$. This produces a quasi-ordered structure called spin wave where the spin orientations at long distances are correlated. When increasing the temperature, the vortex-antivortex pairs tend to occupy larger regions and eventually, when $T \approx T_{KT}$ these regions have a typical diameter comparable to the lattice size and the pairs are decoupled. When considering paths that surround relatively small regions, one observes a proliferation of (anti)vortices. At high temperatures the pairs are completely decoupled and the quasi-ordered spin wave structure is broken, leading to small long range correlations. Fig. 1.4 presents two configurations, extracted above and below the critical temperature and displays what we have discussed so far. For more details on these results and further reading we invite the interested reader to Refs. [2], [6].

## 1.4 Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm is a Monte Carlo technique which is used to sample independent configurations of a system in thermal equilibrium. It is based on a Markovian and homogeneous stochastic process that, starting from any configuration, leads the system to thermal equilibrium with a characteristic time $\tau_r$.

Let $t$ be a discrete time variable labeling the steps of this process, and let $C$ and $C'$ be two configurations of the system. We call $P_t(C)$ the probability that the system is in configuration $C$ at time $t$ and $T(C \to C')$ the transition rate from $C$ to $C'$ (this does not depend on time since the process is supposed to be homogeneous). The Master equation of this process is in general

$$P_{t+1}(C) - P_t(C) = \sum_{C'} \left[ T(C' \to C) P_t(C') - T(C \to C') P_t(C) \right]. \tag{1.28}$$

If the system is in contact with a thermal bath at temperature $T$, then an equilibrium configuration has the canonical probability

$$P_{eq}(C) = \frac{e^{-\beta \mathcal{H}(C)}}{\mathcal{Z}} \tag{1.29}$$

where $\mathcal{H}(C)$ is the Hamiltonian of the system and $\beta = 1/k_B T$.

If we want the system to relax at a stationary equilibrium state when $t \gg \tau_r$, we have to choose the transition rates properly. In the stationary state $P_{t+1}(C) - P_t(C) = 0 \ \forall t$, and if this state is an equilibrium state, then $P_t(C) = P_{eq}(C)$. This is certainly consistent if the transition rates satisfy *detailed balance* for every $C$ and $C'$: from Eqs. 1.28 and 1.29 we get

$$T(C' \to C) P_{eq}(C) = T(C \to C') P_{eq}(C') \tag{1.30}$$

$$\frac{T(C \to C')}{T(C' \to C)} = \exp\left[ -\beta \left( \mathcal{H}(C') - \mathcal{H}(C) \right) \right] = e^{-\beta \Delta \mathcal{H}}. \tag{1.31}$$

The conclusion is that if we choose the transition rate in such a way that it satisfies Eq. 1.31 for every $C$ and $C'$, the Monte Carlo dynamics will relax to equilibrium for sufficiently long times. One of the most common choices is

$$T(C \to C') = \Gamma \min(1, e^{-\beta \Delta \mathcal{H}}), \tag{1.32}$$

which straightforwardly satisfies Eq. 1.31. We remark that this is not the only possible choice, but it is one of the most used in literature, and so we use it throughout this work. For more information and other algorithms we invite the interested reader to Ref. [9].

The general steps one has to make when implementing a Metropolis-Hastings algorithm are the following:

- make a trial move, namely choose a new configuration $C'$ different from $C$ with some rule;
- accept the new configuration (make the system switch to $C'$) with probability

$$P_{accept} = \Gamma^{-1} T(C \to C') = \min(1, e^{-\beta \Delta \mathcal{H}}); \tag{1.33}$$

- repeat these steps untill equilibrium is reached.

It is important to point out that, once equilibrium is reached (say when $t \approx 10\tau_r$), all the configurations generated after that time are all equilibrium states, but they are strongly correlated because the configuration $C'$ is choosen from $C$ with a predefined rule. So, if we want to get statistically independent samples (which is often the case), we have to compute the autocorrelation time $\tau_A$ of the observable $A$ we are interested in, and then choose one sample at least every $\tau_A$ iterations of the process. To this scope we introduce the time autocorrelation function $G_A(\tau)$ of the observable $A$

$$G_A(\tau) = \frac{\langle A_t A_{t+\tau} \rangle - \langle A \rangle^2}{\langle A^2 \rangle - \langle A \rangle^2}, \tag{1.34}$$

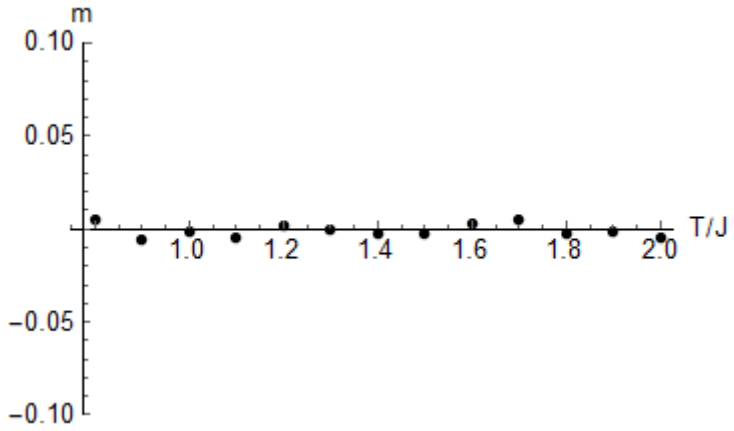



Figure 1.5: Magnetization as a function of temperature at null external magnetic field. Below $T \approx 0.8J$ the finite size effects become relevant and the simulation is no longer reliable since $\xi \gg L$, and hence a fictitious magnetization occurs.

where the expectation value is the average over all times such that $10\tau_r < t < \bar{t} - \tau$ ($\bar{t}$ being the final time at which the process stops). This function is typically an exponential decaying function, so $G_A(\tau) \approx e^{-\tau/\tau_A}$, from which we can compute $\tau_A$.

In some situations the autocorrelation time can be really high and sampling a reasonable number of independent configurations can be computationally challenging: this problem is known as *critical slowing down*, since it is typical of systems near criticality. In some cases, the problem can be solved by smartly choosing the transition rate $T(C \to C')$ (taking into account the constraint in Eq. 1.31) and implementing more sophisticated algorithms.

Let's now discuss how we have implemented the algorithm on the specific cases of the previously discussed models.

**Ising model**

After initializing a random spin configuration $\{S_i\}$ (for example $S_i = +1\ \forall i$), the algorithm consists in the following steps:

- propose a new configuration $C'$ with a trial move, namely flip a randomly chosen spin $S_{i_0}$ (change its sign);
- compute the energy difference between the new and the old spin configurations $\Delta\mathcal{H}$, then accept the new configuration with the probability given in Eq. 1.33.
- repeat these steps $\bar{t}$ times.

Since the Ising model can be exactly solved when $d = 1$ and $d = 2$, we can compare our Monte Carlo simulation with exact resulsts for magnetization and susceptibility.

The simulations of the one dimensional system are performed at temperatures $T > 0.8J$, well above the lower temperature limit stated in Eq. 1.8, which is $0.45J$ for $L = 40$. In this way we are guaranteed that the correlation length is much smaller than the system size $\xi \ll L$. Fig. 1.5 shows that the magnetization is essentialy zero at all the sampling temperatures and the fictitious ordered phase does not appear. In Fig. 1.1a we compare the theoretical and computational results for magnetization as functions of the external field at different temperatures. In Fig. 1.1b we compare the magnetic susceptibility obtained by the simulation with the theoretical result as functions of temperature. Eq. 1.5 shows how one can compute both these quantities.

An analysis of the relaxation characteristic time $\tau_r$ shows that one should discard the first $10L$ Monte Carlo iterations for every sampled temperature for the system to reach equilibrium. Moreover, plotting the autocorrelation function of the magnetization, one can compute $\tau_m$, which depends on temperature and is larger at lower temperatures. Since $\tau_m \approx 100L$ at the lowest sampled temperature $T = 0.8J$,

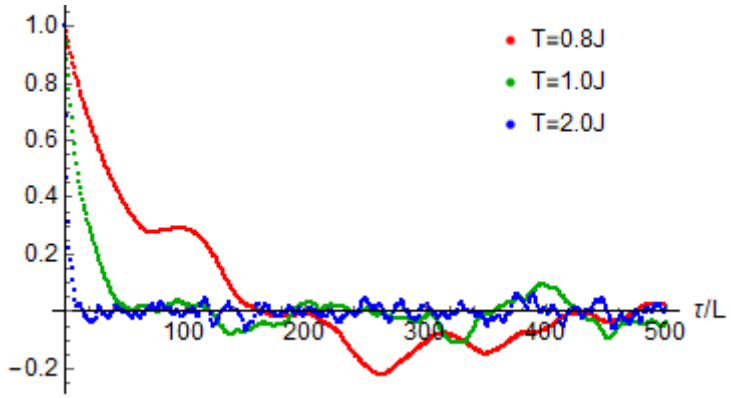



Figure 1.6: Correlation function of the magnetization per site $G_m(\tau)$ as function of Monte Carlo sweeps ($\tau/L$) at different temperatures. The largest correlation time is $\approx 100L$ for the lowest temperature.

as shown in Fig. 1.6, we sampled one spin configuration every $3\tau_m$, so that the expected correlation is lower than 0.05.

For the two dimensional system, the analysis of the characteristic times of the Metropolis-Hastings algorithm shows that $\tau_r \approx 30L^2$, so we sampled after waiting $\approx 100L^2$ steps. The autocorrelation function of magnetization $G_m(\tau)$ decays with $\tau_m \approx 15L^2$ at temperatures far from the critical temperature, but the decay is slowed down at the critial point and $\tau_m \approx 200L^2$. This is the already mentioned critical slowing down, which implies great computational cost, especially for large lattices. A compromise between statistical independence of data and reasonable computational time is needed: in our case, we choose to sample one configuration every $150L^2$. This choice is maintained throughout this work, independently from temperature and from lattice size. Another possible solution to the problem would be implementing a different Monte Carlo algorithm known as *Wolff algorithm*. It basically consists in changing the way we propose the new configuration, in particular flipping a randomly chosen set of neighbouring spins sharing the same value of the spin. The implementation of this algorithm is beyond the scope of this work. The interested reader is invited to Ref. [9].

**Potts model**

In this case the basic conclusions are the same as for the Ising model. The only difference is the trial move that now consists in randomly changing the state of a given site. The new state is uniformly and randomly chosen amongst the other $q-1$ states.

**XY model**

Equilibrium configurations of the XY model can be sampled using the Metropolis-Hastings algorithm, with a random change of the spin angle of a randomly selected site as a trial move (in place of the spin flip). Despite its simple implementation, the algorithm suffers of a high autocorrelation time for the relevant observables, expecially at low temperatures and this makes it computationally inefficient. As a good compromise between reasonable computational time and statistical independence of the dataset, we choose a relatively small lattice size $L = 20$ and a sampling time $t = 400L^2$ in the Monte-Carlo dynamics.

For the purposes of this work it is interesting to generate different configurations with local vorticities as features. The idea is to consider a new square lattice made by the centers of all the squares with unitary size (plaquettes). The local vorticity $q_{ij}$ associated to site $i, j$ in the new lattice can be computed with the discretized vertion of the integral in Eq. 1.27 using the four angles at the vertices of the plaquette:

$$q_{ij} = \frac{1}{2\pi}\left[f(\theta_{i+1,j} - \theta_{ij}) + f(\theta_{i+1,j+1} - \theta_{i+1,j}) + f(\theta_{i,j+1} - \theta_{i+1,j+1}) + f(\theta_{ij} - \theta_{i,j+1})\right], \qquad (1.35)$$

where $f(x)$ is a sawtooth function that returns the value of the difference between angles $x \in [-2\pi, 2\pi]$

rescaled in the interval $[-\pi, \pi]$:

$$f(x) = \begin{cases} x - 2\pi & \text{if} \quad x > \pi \\ x & \text{if} \quad -\pi \leq x \leq \pi \\ x + 2\pi & \text{if} \quad x < -\pi \end{cases} . \tag{1.36}$$

Starting from a $L \times L$ lattice, the new lattice has $(L-1) \times (L-1)$ sites. In order to obtain a $L \times L$ lattice, we can start from a $(L+1) \times (L+1)$ lattice, where $\theta_{L+1,j}$ and $\theta_{i,L+1}$ are chosen depending on the choice of boundary conditions. Since we have sampled the XY model with periodic boundary conditions, we made the following identifications $\theta_{L+1,j} = \theta_{1,j}$ and $\theta_{i,L+1} = \theta_{i,1}$. The set of values $\{q_{ij}\}$ obtained with this procedure is called vorticity dataset in the following.

# Chapter 2

# Design of spin models

In this chapter we discuss the problem of model reconstruction from data using some basic tools of machine learning. Given a set of configurations of a system, with their relative energies, we try to design the simplest Hamiltonian compatible with the data. To this scope, the starting point is making an educated guess on the symmetry properties of the system, so that the general analytic structure of the Hamiltonian is known a priori. Another important result we want to achieve is predicting the energy of unobserved configurations after a training procedure, without overfitting, which is the central problem of *supervised machine learning*. In the first part of the chapter we discuss the penalized linear regression in supervised learning, which is the basic tool to address our physical problem, and we apply it to the Ising model. The last part is dedicated to some examples of model reconstruction.

## 2.1 Linear regression

Let's consider a physical system characterized by some features represented by a row vector $\mathbf{x}^T$ and a relative scalar observable $y$. The components of $\mathbf{x}^T$ are labeled by an index $j$ that runs from 1 to $p$, where $p$ is the number of measurable features. Let's suppose we have the prior knowledge that $\mathbf{x}^T$ and $y$ are linearly related via a set of coupling constants $\mathbf{w}$, namely

$$y = \mathbf{x}^T \cdot \mathbf{w} \tag{2.1}$$

If we have $n$ observations $\{\mathbf{x}_i^T, y_i\}_{i=1,\dots,n}$, we can define a configuration matrix $\mathbf{X}$ whose rows are $\mathbf{x}_i^T$ and a vector $\mathbf{y}$ whose elements are $y_i$. Then the linear relation generalizes to

$$\mathbf{y} = \mathbf{X}\mathbf{w} \tag{2.2}$$

The task of linear regression in machine learning is to train an algorithm using a training dataset $\{\mathbf{X}_{train}, \mathbf{y}_{train}\}$ to optimize the parameters $\mathbf{w}$ in such a way that we can correctly predict the values of $\mathbf{y}_{test}$ of a new unseen test dataset $\mathbf{X}_{test}$.

There is a subtle difference between fitting and learning which is worth remarking. Fitting means finding the best parameters that explain the observed behaviour of the training samples. Learning means using the training dataset to find the best parameters that allow good predictions on new, unseen data. Since the number of parameters we have to determine ($p$) is in practice very large, sometimes even greater than the number of samples $p > n$, there is a risk of overfitting the given data, with loss of predicting power. Overfitting means finding parameters that perfectly match the training data, making wrong predictions on unseen data.

The optimal parameters are obtained by minimizing a properly chosen cost function $C(\mathbf{w})$, that describes the difference from observed and predicted results. The simplest cost function we can think of is the square norm of the difference between prdicted and measured data

$$C(\mathbf{w}) = \frac{1}{2} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 . \tag{2.3}$$

In this context $\|v\|_k = \left(\sum_i v_i^k\right)^{1/k}$ is the $k$-norm of a vector, and when $k$ is not specified it is tacitly $k = 2$. Minimizing this function consists in performing the usual linear fit, with a correlated risk of overfitting. The risk of overfitting is reduced adding suitable terms (penalization terms) to the cost function.

### 2.1.1 $L_2$ penalized regression

The $L_2$ penalized regression (or Ridge regression) consists in choosing the coupling constants that minimize the $L_2$-penalized cost function

$$C_\lambda(\mathbf{w}) = \frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \lambda\|\mathbf{w}\|^2. \tag{2.4}$$

The minimum of this function is found by solving $\nabla C(\mathbf{w}) = 0$ for $\mathbf{w}$, that after straightforward calculations is given by

$$\mathbf{w}_{Ridge}(\lambda) = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbb{1}\right)^{-1}\mathbf{X}^T\mathbf{y}, \tag{2.5}$$

where $\mathbb{1}$ is the $p \times p$ identity matrix.

The cost function in Eq. 2.4 comes from Bayes' theorem. Let $P(\mathbf{X}, \mathbf{y}|\mathbf{w})$ be the probability density function of observing the configuration $\{\mathbf{X}, \mathbf{y}\}$ given the parameters $\mathbf{w}$; let $P(\mathbf{w})$ be the prior probability density function of the parameters and finally let $P(\mathbf{w}|\{\mathbf{X}, \mathbf{y}\})$ be the likelihood (i.e. the probability that the coupling constants are $\mathbf{w}$, once we know the data are $\{\mathbf{X}, \mathbf{y}\}$). If observations are statistically independent, then $P(\mathbf{X}, \mathbf{y}|\mathbf{w}) = \prod_i P(\mathbf{x}_i^T, y_i|\mathbf{w})$; moreover we can assume that they are gaussian distributed with variance $\sigma^2$

$$P(\mathbf{x}_i^T, y_i|\mathbf{w}) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{1}{2\sigma^2}\left(y_i - \mathbf{x}_i^T \cdot \mathbf{w}\right)^2\right]. \tag{2.6}$$

If we can reasonably expect most of our parameters to be zero and all statistically independent, then we can assume as a prior knowledge that $P(\mathbf{w}) = \prod_i P(w_i)$ and that they are normally distributed with mean 0 and variance $\tau^2$

$$P(w_i) = \frac{1}{\sqrt{2\pi\tau^2}}\exp\left[-\frac{1}{2\tau^2}w_i^2\right] \tag{2.7}$$

Bayes' theorem states that

$$P(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \frac{P(\mathbf{X}, \mathbf{y}|\mathbf{w})P(\mathbf{w})}{P(\mathbf{X}, \mathbf{y})} \tag{2.8}$$

and it can be used to estimate the log-likelihood $\mathcal{L}(\mathbf{w}) = \log P(\mathbf{w}|\mathbf{X}, \mathbf{y})$, which after some manipulations is

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{\sigma^2}C_\lambda(\mathbf{w}) + \text{const.} \tag{2.9}$$

where $\lambda = \sigma^2/\tau^2$ is the variances ratio, $C_\lambda(\mathbf{w})$ is the cost function introduced in Eq. 2.4, and the constant does not depend on $\mathbf{w}$. We notice that $\lambda > 0$ and that in the case $\lambda \gg 1$ the prior on parameters is strongly peaked at $\mathbf{w} = 0$. In conclusion, minimizing the cost function (Eq. 2.4) is the same problem than maximizing the log-likelihood (Eq. 2.9).

Once we find the Ridge estimate for the parameters we need an estimator for the performance of our algorithm. The performance of course depends on $\lambda$, which is an hyperparameter (a parameter to be found a posteriori checking the most performing value). The process of choosing the best value for $\lambda$ is called *validation*, and is the following. The available data are divided into two classes: $\bar{n}$ training data for estimating $\mathbf{w}_{Ridge}$ and $n - \bar{n}$ testing data, for evaluating the out of sample performance of the algorithm. Using the training outcome for parameters and either $\mathbf{X}^{train}$ or $\mathbf{X}^{test}$, one can give a prediction $\mathbf{y}^{pred} = \mathbf{X}\mathbf{w}_{Ridge}$ and compare it to the known $\mathbf{y}^{true}$. The coefficient of performance is given by (Ref. [7])

$$R^2(\lambda) = 1 - \frac{\sum_i \left|y_i^{pred} - y_i^{true}\right|^2}{\sum_i \left|y_i^{pred} - \frac{1}{\bar{n}}\sum_k y_k^{true}\right|^2}. \tag{2.10}$$

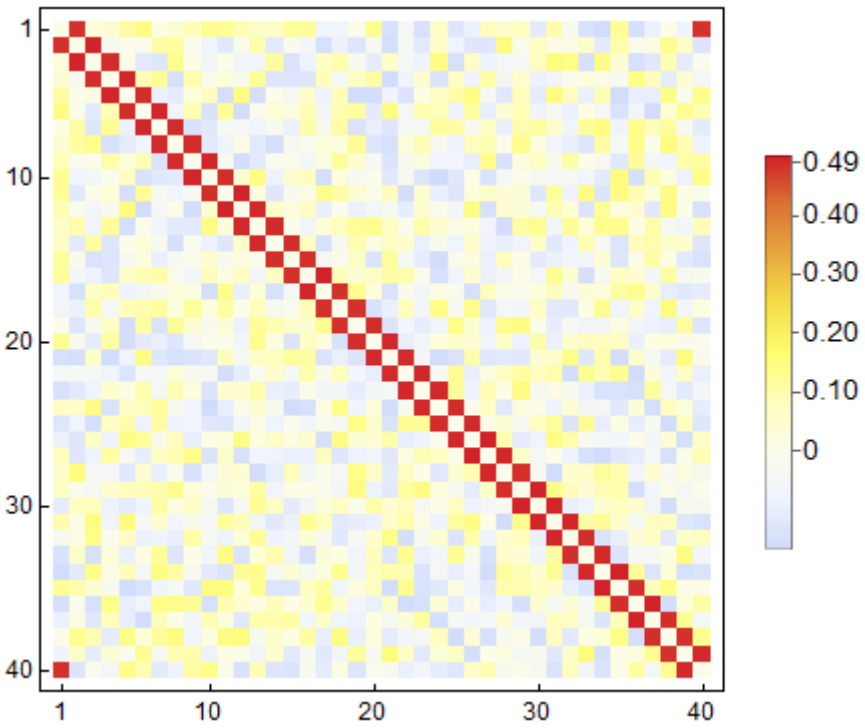



Figure 2.1: Representation of the interaction matrix $J_{ij}$ learnt by Ridge regression with $\bar{n} = 5000$ and $\lambda = 0.01$. It turns out that $J_{i,i+1} \approx J_{i,i-1} \approx 0.5$, and also $J_{1,L} \approx J_{L,1} \approx 0.5$ because of periodic boundary conditions, while all the other coefficients are much smaller.

Despite the square, this quantity is not guaranteed to be positive, and in the following we will discuss an example where it is not. When $\mathbf{y}^{pred}$ is predicted using the training set, $R^2$ is an index of fitting performance; when it is predicted using the test set, $R^2$ estimates predicting performance on new data. The best estimate for the hyperparameter $\lambda$ is the one that maximizes $R^2(\lambda)$ of the test set.

For a one dimensional Ising model with no external fields, Ridge regression can be applied. Supposing we don't know that data were generated by nearest neighbors coupling, we could in principle guess a two body Hamiltonian of the form $\mathcal{H} = -\sum_{\mu\nu} J_{\mu\nu} S_\mu S_\nu$ (a priori excluding three body interactions, external fields etc.). The available data are $n$ spin configurations (a set of $L$ spins) and the corresponding energies $E$; the configurations are randomly extracted from the phase space (we haven't performed a Metropolis-Hastings sampling). In order to obtain a linear relation as Eq. 2.2, we have to introduce an index $j = (\mu - 1)L + \nu$ running from 1 to $p = L^2$ and call $x_j = -S_\mu S_\nu$, $w_j = J_{\mu\nu}$ and $y = E$.

With these prescriptions we performed a Ridge regression on the one dimensional Ising model, using 5000 training samples and 1000 test samples with $L = 40$ and $J = 1$. The tuneable parameter $\lambda$ was set to 0.01 after the validation process shown in Fig. 2.2. In Fig. 2.1 we present the learnt weights $\mathbf{w}$ representing the coupling constants, organized in a $L \times L$ matrix with a suitable color scheme. What emerges from the figure is that a large number of parameters that should vanish in principle is instead not zero, even if it's small. Anyway the most relevant parameters are clearly $J_{i,i+1}$ and $J_{i-1,i}$, both being approximately 0.5 for every $i$. From this example, two features of Ridge regression emerge: firstly it learns symmetric weights, equally dividing the interaction energy between $J_{ij}$ and $J_{ji}$; secondly it tries to keep as much parameters as possible and it assigns small values rather than zero. In Fig. 2.2 we show the coefficient of performance $R^2(\lambda)$ as a function of $\lambda$, ranging from $10^{-4}$ to $10^5$, in each step increasing $\lambda$ of a factor 10. We checked the performance both with the training and test sets and compare the results to another regression with $\bar{n} = 500$. As expected, for large $\lambda$ the performance decreases, since the parameters are more likely to vanish. Indeed, when $\lambda \gg 1$, then $\tau \ll \sigma$, meaning that we have a strong prior that all the parameters should approximately vanish. At small $\lambda$ the performance of the training set is $\approx 100\%$ even with a small training sample, meaning that the model fits the data. The performance of the test set is $\approx 100\%$ with a large training set,

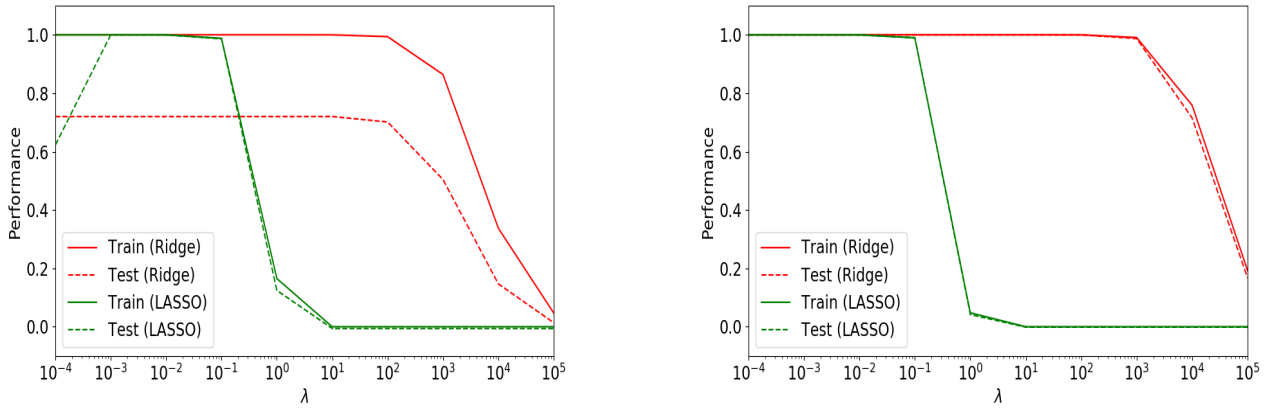



Figure 2.2: $R^2(\lambda)$ performance coefficient as a function of the hyperparameter $\lambda$ when performing Ridge or LASSO regression with guessed two body Hamiltonian and one dimensional Ising dataset with $L = 40$. In the left panel we have used $\bar{n} = 500$ training samples; in the right panel we have used $\bar{n} = 5000$ training samples. For small enough $\lambda$ the performance is good both when computed to the train and test sets, meaning that the model is not overfitting and is likely to be the right underlying model.

but even with a small training set it is $\approx 70\%$. So we can conclude that not only the model fits the training, but it predicts correctly the test, so there is no overfitting.

As a final remark we underline that not only Ridge regression avoids overfitting in machine learning problems, but also provides a good alternative to simple linear regression when $\mathbf{X}^T\mathbf{X}$ has no inverse. Minimizing the ordinary least squares cost function of Eq. 2.3 we get

$$\mathbf{w}_{OLS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{w}_{Ridge}(0), \tag{2.11}$$

that indeed requires inverting $\mathbf{X}^T\mathbf{X}$. It is easy to prove that the regularized matrix $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ can always be inverted for suitable $\lambda$, so that Ridge regression can be performed in any case.

### 2.1.2 $L_1$ penalized regression

Let's consider the same linear problem of Eq. 2.2. $L_1$ penalized regression (or LASSO, acronym of Least Absolute Shrinkage and Selection Operator) consists in minimizing the cost function

$$C_\lambda(\mathbf{w}) = \frac{1}{2}\left\|\mathbf{X}\mathbf{w} - \mathbf{y}\right\|_2^2 + \lambda\left\|\mathbf{w}\right\|_1, \tag{2.12}$$

where $\|\cdot\|_1$ is the standard $L_1$ norm, in particular $\|\mathbf{w}\|_1 = \sum_{j=1}^{p}|w_j|$. We point out that this function is convex but non differentiable when one of the parameters is zero.

This choice of the cost function in Eq. 2.12 consists in chosing a different, non gaussian, prior probability density function for the weights

$$P(w_i) = \frac{1}{2k}e^{-k|w_i|}, \tag{2.13}$$

where $k > 0$, while keeping a gaussian function $P(\mathbf{X}, \mathbf{y}|\mathbf{w})$ as in Eq. 2.6. With this choice one assumes that $w_i = 0$ is more likely than in the case of Ridge penalization and this is the reason why LASSO is more used for sparse regression (i.e. in systems where we know that only few parameters are significant). Following the same steps as in the Ridge regression, we can find the log-likelihood for LASSO:

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{\sigma^2}C_\lambda(\mathbf{w}) + \text{const.} \tag{2.14}$$

where now $C_\lambda$ is the cost function of Eq. 2.12 and $\lambda = k\sigma^2$. Thus, again, minimizing the cost function is the same problem as maximizing the log-likelihood.

### 2.1.3 Proximal algorithm for LASSO

In this section we show that the LASSO problem can be mapped into a fixed-point problem, which can be solved iteratively, thus providing an algorithm to minimize the cost function. Since $C_\lambda(\mathbf{w})$ is a convex function, it has a global minimum, but since the cost function is not differentiable, the global minimum doesn't necessarily make the gradient $\nabla C_\lambda(\mathbf{w})$ vanish. In practice, the gradient is not even defined at all points, so we should generalize it to the so called *subgradient*.

Let's begin extending the concept of gradient to non differentiable convex functions. The subgradient of a convex function $f(\mathbf{w})$ at a given point $\mathbf{w}_0$ is any vector $\mathbf{v}$ such that

$$f(\mathbf{w}) - f(\mathbf{w}_0) \geq \mathbf{v} \cdot (\mathbf{w} - \mathbf{w}_0) \ \forall \mathbf{w} \tag{2.15}$$

The subgradient is not unique in general, unless the function is differentiable at $\mathbf{w}_0$, and in that case the only subgradient is the gradient itself: $\mathbf{v} = \nabla f(\mathbf{w}_0)$. The set of all subgradients at a given point is called subdifferential and denoted with $\partial f(\mathbf{w}_0)$. It is possible to prove that $\mathbf{w}_0$ is a global minimum for the function $f$ if and only if $0 \in \partial f(\mathbf{w}_0)$. An interesting example is the subderivative of $f(w) = |w|$ at $w_0 = 0$, which is $\partial f(0) = [-1, 1]$.

Let's now define the proximal operator of a convex function $f$:

$$\text{prox}_f(u) = \arg\min_{\mathbf{w}} f(\mathbf{w}) + \frac{1}{2} \|\mathbf{w} - \mathbf{u}\|_2^2 \tag{2.16}$$

As a useful example, let's consider a function $\phi(w) = \lambda|w|$. The proximal operator for this function is

$$\text{prox}_\phi(u) \equiv S_\lambda(u) = \begin{cases} u - \lambda & \text{if } u > \lambda \\ 0 & \text{if } -\lambda \leq u \leq \lambda \\ u + \lambda & \text{if } u < -\lambda \end{cases}, \tag{2.17}$$

which is called *soft thresholding operator*. The most relevant property of this operator is that $\mathbf{w}_0$ is a global minimum for the function $f$ if and only if it is a fixed point of $\text{prox}_f$, namely $\mathbf{w}_0 = \text{prox}_f(\mathbf{w}_0)$. This property is quite simple to prove and puts the problem under another perspective: we now have to find the fixed point of the proximal operator.

It is possible to prove that the following iterative procedure converges to the searched fixed point, and hence to the minimum of $f$:

$$\mathbf{w}^{(k+1)} = \text{prox}_{tf}(\mathbf{w}^{(k)}) \tag{2.18}$$

where $t > 0$ has to be choosen properly. If the target function $f$ is sum of a differentiable convex function $g$ and a non differentiable convex one $h$, one can start with an arbitrary chosen value $\mathbf{w}^{(0)}$ and then iteratively apply

$$\mathbf{w}^{(k+1)} = \text{prox}_{th}(\mathbf{w}^{(k)} - t\nabla g(\mathbf{w}^{(k)})) \tag{2.19}$$

where $t > 0$. This method is called proximal gradient method and it is possible to prove that it does converge to the fixed point, namely $\left\|\mathbf{w}^{(k)} - \mathbf{w}_0\right\|_2 \to 0$ when $k \to \infty$. The proof is the following:

$$\begin{aligned} \mathbf{w}^{k+1} &= \text{prox}_{th}(\mathbf{w}^{(k)} - t\nabla g(\mathbf{w}^{(k)})) \\ &= \arg\min_{\mathbf{w}} \left[ th(\mathbf{w}) + \tfrac{1}{2} \left\|\mathbf{w} - \mathbf{w}^{(k)} + t\nabla g(\mathbf{w}^{(k)})\right\|_2^2 \right] \\ &= \arg\min_{\mathbf{w}} \left[ th(\mathbf{w}) + tg(\mathbf{w}^{(k)}) + t\nabla g(\mathbf{w}^{(k)}) \cdot (\mathbf{w} - \mathbf{w}^{(k)}) + \tfrac{1}{2} \left\|\mathbf{w} - \mathbf{w}^{(k)}\right\|_2^2 \right] \\ &\approx \arg\min_{\mathbf{w}} \left[ tf(\mathbf{w}) + \tfrac{1}{2} \left\|\mathbf{w} - \mathbf{w}^{(k)}\right\|_2^2 \right] \\ &= \text{prox}_{tf}(\mathbf{w}^{(k)}) \end{aligned} \tag{2.20}$$

where in the third row we have subtracted and added proper constants (terms that don't depend on $\mathbf{w}$).

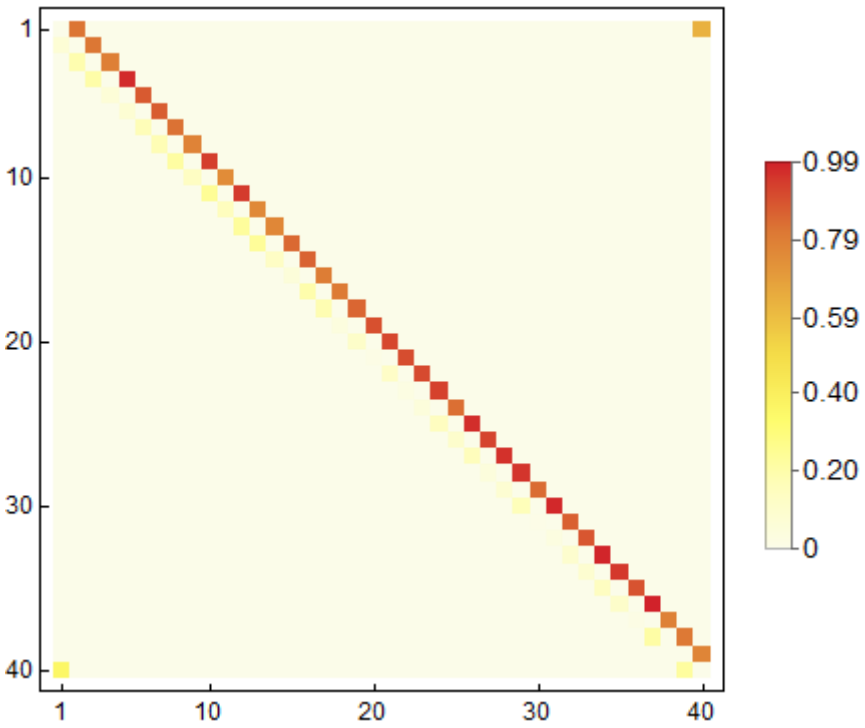



Figure 2.3: $J_{ij}$ for a one dimensional Ising model of size $L = 40$. The LASSO algorithm learns asymmetric parameters: $J_{i,i+1} \approx 1$ and $J_{1,L} \approx 1$, while the others are approximately zero. In the simulation we used $\bar{n} = 5000$, $\lambda = 0.01$.

We now have to compute the proximal operator for the LASSO cost function (2.12). Taking as $g$ the ordinary least square function and as $h$ the $L_1$ penalization term, we get the following simple algorithm, which is called iterative soft thresholding algorithm (ISTA)

$$\mathbf{w}^{(k+1)} = S_{t\lambda}\left(\mathbf{w}^{(k)} + tX^T(\mathbf{y} - X\mathbf{w}^{(k)})\right) \tag{2.21}$$

The interested reader can find more details on this topic, as well as the omitted proofs in Refs. [22,23].

Setting up the Ising learning as a linear problem, as shown in the previous section, we can now find the parameters with LASSO. The training set is made by 5000 samples, while the test set has 1000 samples, each with lattice size $L = 40$ and coupling constant $J = 1$. After studying the pefromance as a function of $\lambda$ (Fig. 2.2), we chose $\lambda = 0.01$. The learnt parameters are shown in Fig. 2.3, that should be compared to the results of Ridge regression of Fig. 2.1. First we observe that in this case the weights are learnt asymmetrically, since $J_{i,i+1} \approx 1$ and $J_{i,i-1} \approx 0$ for every $i$. Secondly, as expected from the theoretical discussion, many learnt weights are exactly zero, so LASSO pefroms a feature selection and tries to keep the smallest possible subset of parameters.

In conclusion, looking at Fig. 2.2 we can conclude that, as in the case of Ridge regression, LASSO does not overfit, even if the training set is small. It is interesting to notice that when $\bar{n}$ is small, the performance of the test set decreases at very small $\lambda$, since features are unlikely to vanish (even if their theoretical value is zero).

## 2.2 Accepting or rejecting models

A key question we have posed in this chapter is whether or not one can use penalized linear regression to find the underlying physical model of given data. We have pointed out that the first step is to guess the basic structure of the model, and in particular the Hamiltonian as a function of the model's features. Of course when absolutely no prior knowledge on the system is available, one is induced to make the simplest guesses for the Hamiltonian.

In this spirit, when spin configurations and their energies are given, the two body interaction Hamiltonian $\mathcal{H} = -\sum_{ij} J_{ij} S_i S_j$ is not the simplest possibility. In principle one could assume that all spins

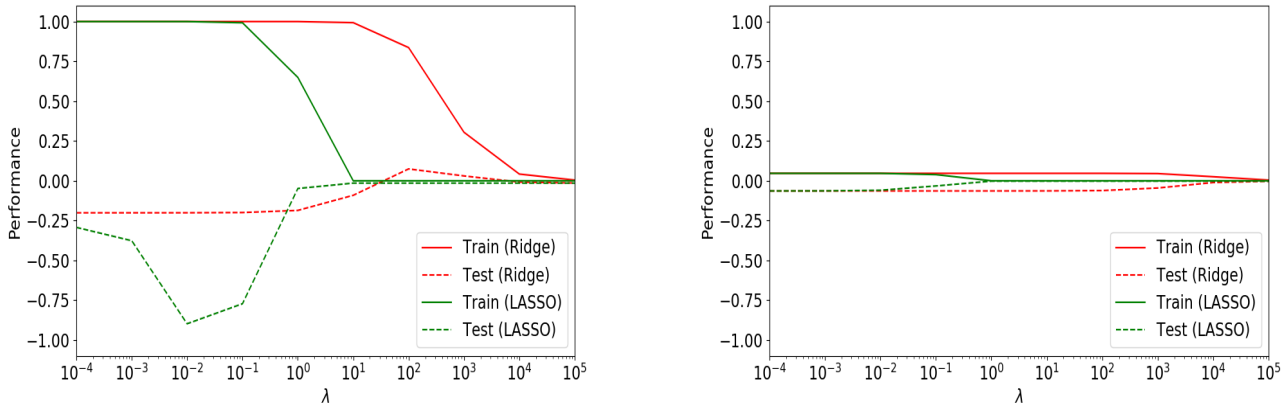



Figure 2.4: $R^2(\lambda)$ performance coefficient as a function of the hyperparameter $\lambda$ when performing Ridge or LASSO regression with guessed Hamiltonian 2.22 and one dimensional Ising dataset with $L = 200$. In the left panel we have used $\bar{n} = 200$ training samples; in the right panel we have used $\bar{n} = 5000$ training samples. In the former case there is a clear overfitting that would not be clear without performing a validation process; in the latter the guessed model doesn't even fit the training.

are not interacting, and that they are coupled to an external field, with the one body Hamiltonian

$$\mathcal{H} = \sum_i h_i S_i. \tag{2.22}$$

In this section we apply both Ridge and LASSO regressions to the Ising model dataset, sampled by the Hamiltonian 1.1, assuming that the underlying model is given by Eq. 2.22. In this case energy is linearly related to the spins, so the design matrix $\mathbf{X}$ is a $n \times L$ matrix with spin configurations in its rows, while the weights vector has the $L$ external fields $h_i$ as components $\mathbf{w}_i = h_i$.

In this case the validation process is crucial for deciding whether the proposed model is suitable or not. In Fig. 2.4 we present the performance coefficient $R^2(\lambda)$ in two cases: when the number of training samples is much bigger than the number of parameters $\bar{n} \gg L$ (right panel) and when they have the same value $\bar{n} = L$ (left panel) using both the training and test set to compute the predicted energy $\mathbf{y}^{pred.}$. When $\bar{n} \gg L$ the failure of the proposed model is clear, since the performance vanishes for every chosen $\lambda$ even when calculated with the training set, meaning that the model doesn't even fit the observed data. When $\bar{n} \approx L$, the performance computed with the training set is good for small $\lambda$, even if the model is not correct. Looking at the performance computed with the test set though, it is clear that this is a case of overfitting, since $R^2(\lambda)$ is small (or negative) for every $\lambda$. In this case the validation process has proved to be crucial for rejecting the proposed model, that without validation would have been accepted.

## 2.3 Learning non standard Ising models

In the previous sections we have tested the supervised learning on the standard Ising model, namely taking $J_{ij} = J$ for nearest neighbors sites $i$ and $j$ and zero otherwise; but we have remarked the generality of the methods, suggesting that it could be used for a general $J_{ij}$. In order to check this out we applied Ridge regression to data generated from eight different non standard Ising Hamiltonians and we tried to rebuild this Hamiltonians *a posteriori*. In other words we tried to guess an unknown Hamiltonian starting from some configurations and their energy.

We generated every spin model in a one dimensional lattice of size $L = 100$, randomly extracting configurations from the phase space and then we applied Ridge regression with $N_{train} = 7000$ training data and $N_{test} = 1000$ test data; we then plotted the optimized parameters as in Fig. 2.5. In order to provide a better picture of the model we then plotted $J(|i-j|)$, which gives the interaction energy at every distance between site $i$ and site $j$. It is important to remark that $J(|i-j|)$ is not a function

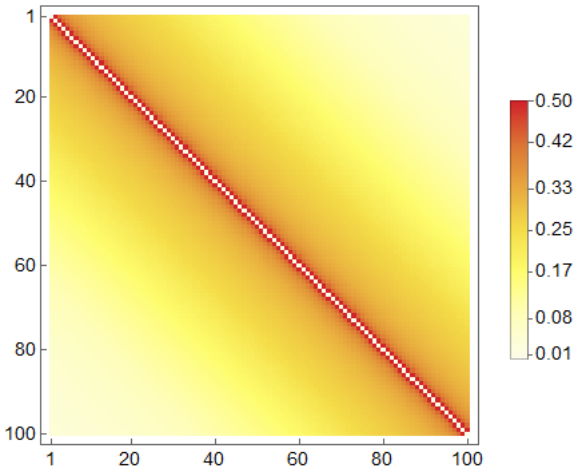
(it can assume different values at the same distance $|i-j|$), since in principle there is no translational symmetry in the system. The hyperparameter $\lambda$ has been chosen through a validation process, in order to maximize the performance on the test set.

In Figs. 2.5, 2.6 we present the results of the procedure described above. From these figures we can provide a good ansatz on the learned models:

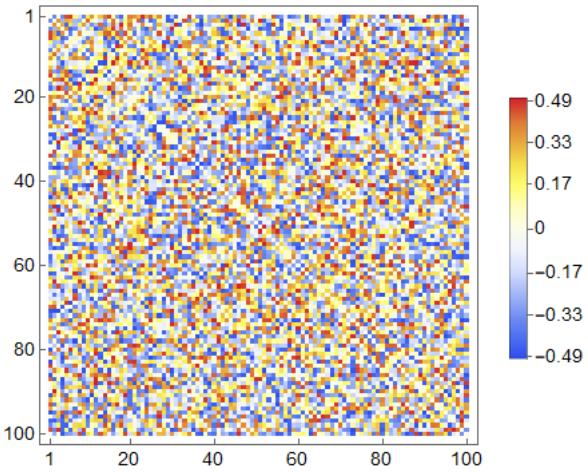
- a), c), f) and h) are spin models with long range interactions. In f) the coupling constants are constant with distance; in the other cases they decay as power laws $J(r) = Ar^{-\alpha}$ (when plotted in a logarithmic scale in both axes $J(r)$ is clearly linear). After fitting one gets $A = 0.5$ in all cases, while the exponent is $\alpha = 0.5, 2.0, 1.5$ in $a), c), h)$ respectively.
- b), e) and g) are spin models with randomly selected interactions. In particular, in c) the interactions are long range, while in the other cases they are only amongst nearest neighbours (only ferromagnetic in g); both ferromagnetic and antiferromagnetic in e)). Plotting the probability distribution of the coupling constants on a histogram shows that the most likely probability density functions from which these constants have been extracted are uniform functions.
- d) is the usual antiferromagnetic Ising model with a negative coupling constant between nearest neighbours and no interaction between further sites.

In conclusion we observe that a penalized linear regression can be used to deduce the coupling constants of any model, once one can find a suitable input that is linearly related to the output energy. In particular not only we have distinguished long range from short range, ferromagnetic from antiferromagnetic interacting systems, but we have also provided a good estimate for the parameters of $J(r)$.

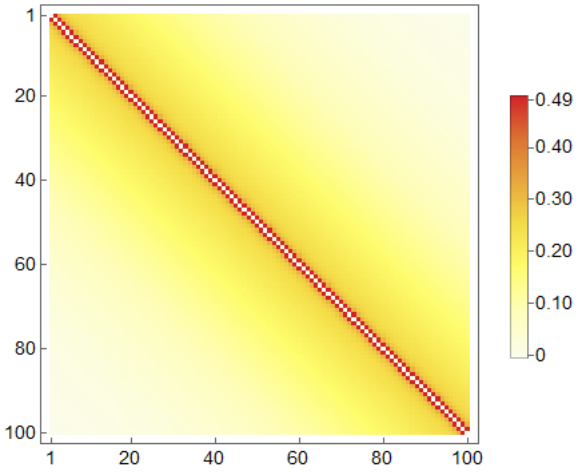
(a)


1
20
40
60
80
100
0.50
0.42
0.33
0.25
0.17
0.08
0.01

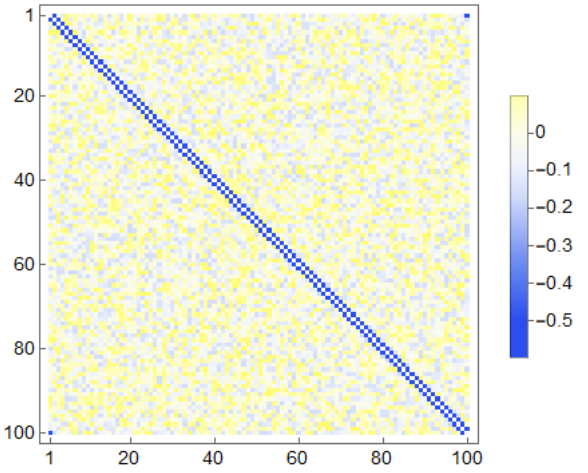
(b)


1
20
40
60
80
100
0.49
0.33
0.17
0
−0.17
−0.33
−0.49

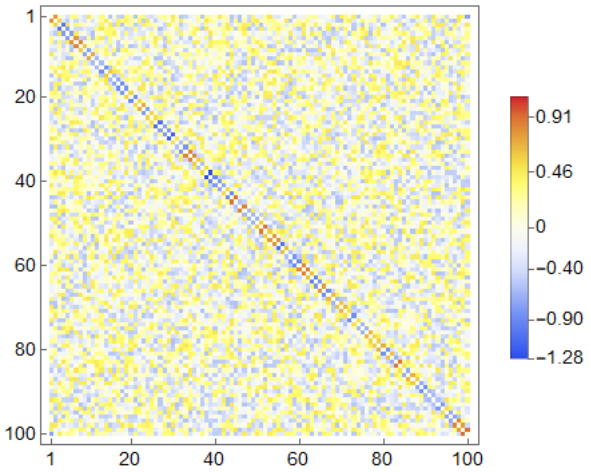
(c)


1
20
40
60
80
100
0.49
0.40
0.30
0.20
0.10
0

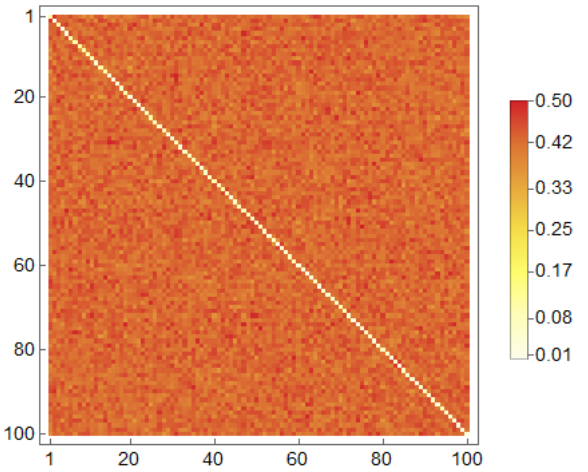
(d)


1
20
40
60
80
100
0
−0.1
−0.2
−0.3
−0.4
−0.5

(e)


1
20
40
60
80
100
0.91
0.46
0
−0.40
−0.90
−1.28

(f)


1
20
40
60
80
100
0.50
0.42
0.33
0.25
0.17
0.08
0.01

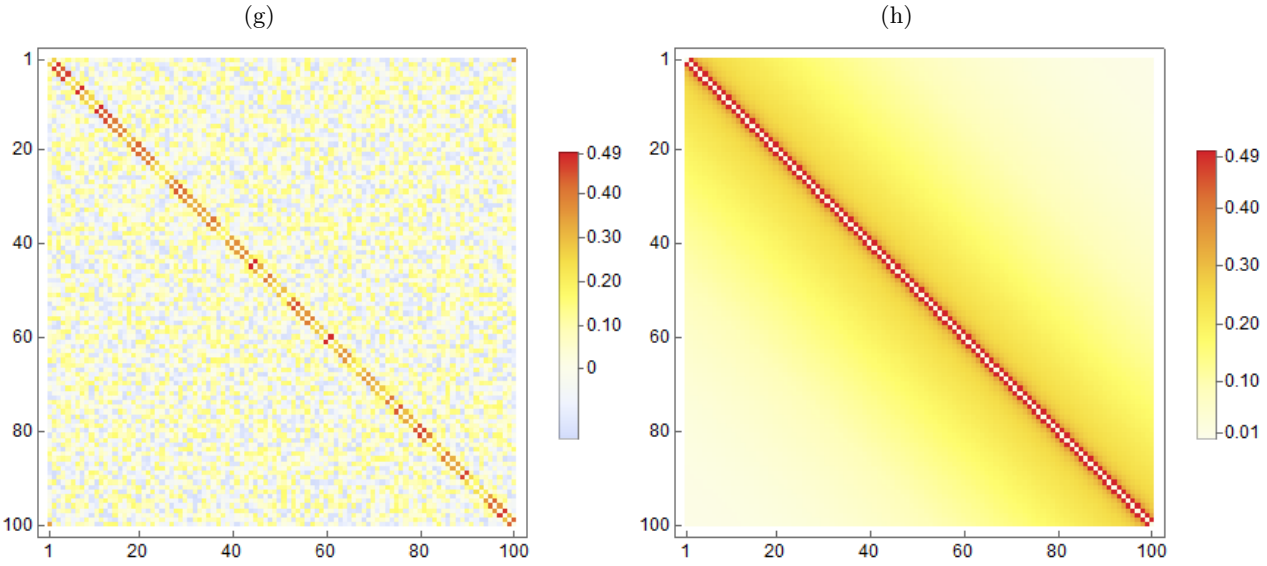
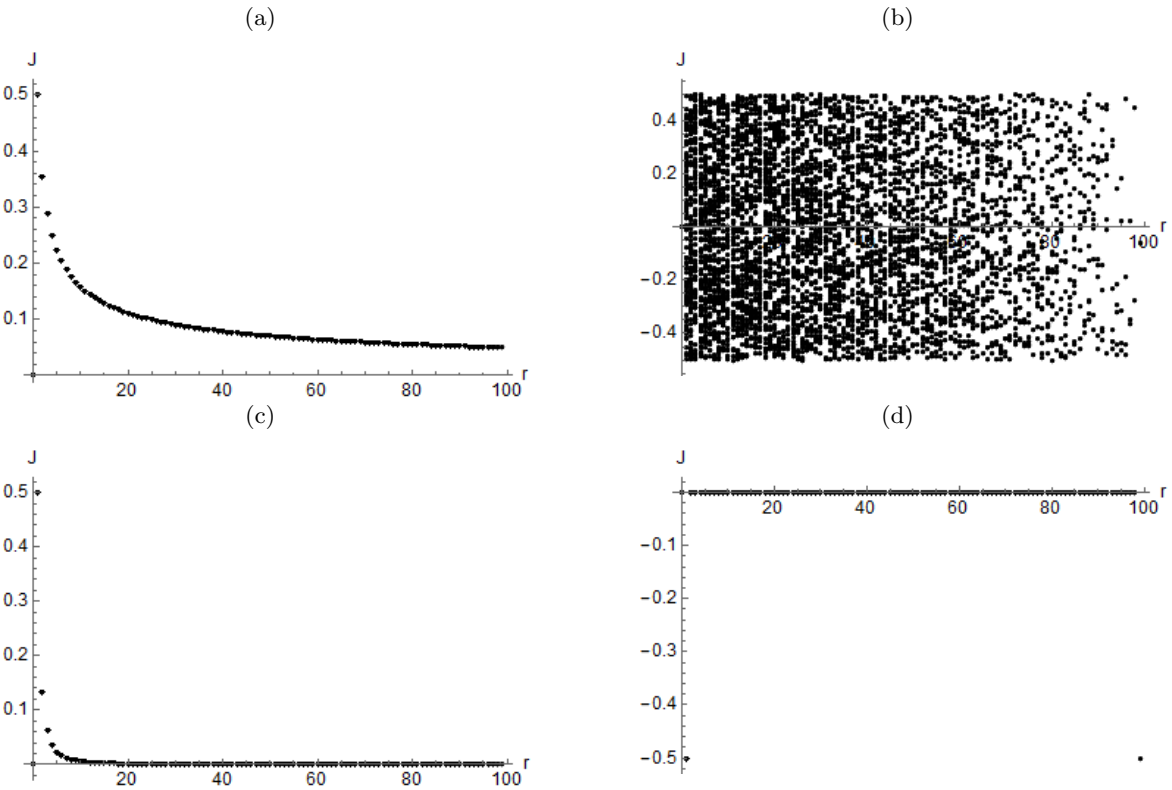



Figure 2.5: Picture of the coupling coefficients $J_{ij}$ of the guessed two body Hamiltonian as a function of the lattice sites $i$ and $j$. Each figure represents the results of Ridge regression on different datasets sampled with unknown models.



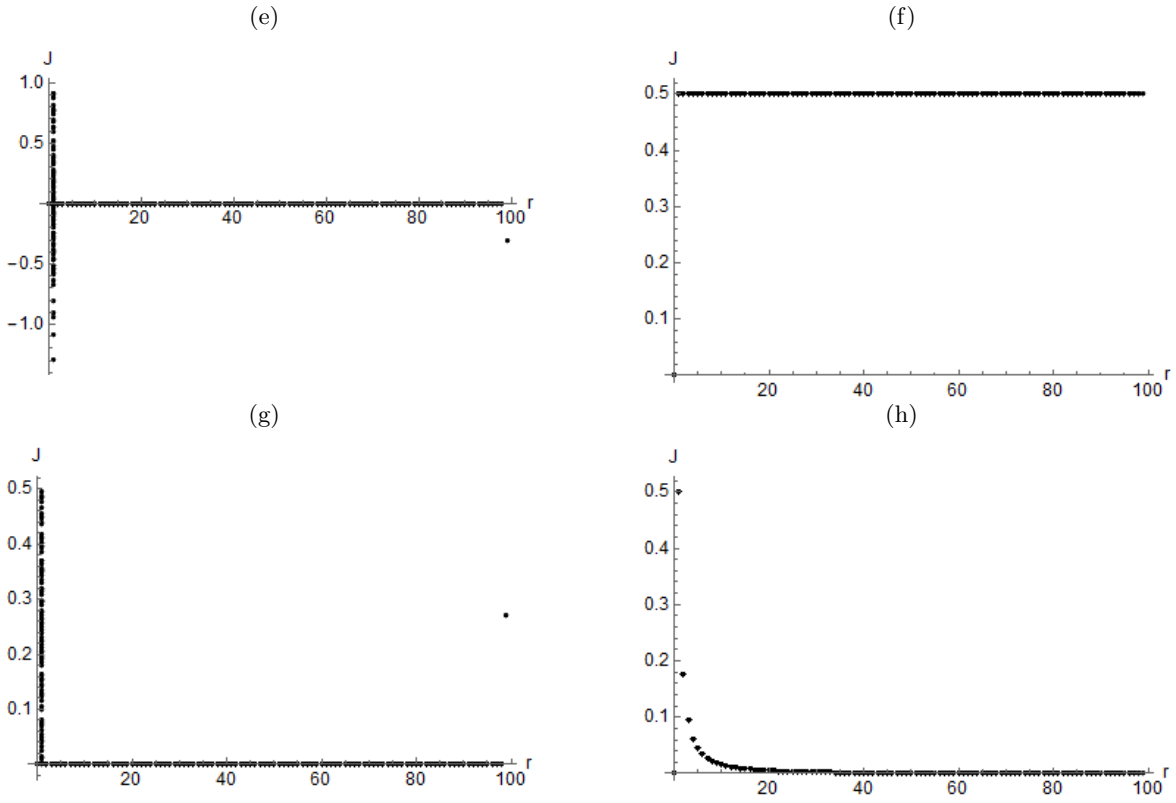



Figure 2.6: Dependence of the coupling constant $J$ on the distance $r$ between site $i$ and $j$ in the lattice. The eight panels a), b), ..., g) are referred to the respective models of Fig. 2.5.

# Chapter 3

# Phase classification with supervised learning

In this chapter we discuss algorithms of supervised machine learning for phase classification of spin models. In particular we start discussing the basic idea of softmax regression for classification of data into a defined number of classes and then we generalize it to the more sophisticated structure of neural networks. We apply these tools to classify the different phases of spin models, discussing how a neural network learns their differences. The interest of this study relies on the capability of machine learning to generalize beyond the observed training dataset. As shown in Ref. [34], a neural network trained with a standard Ising model can correctly predict features of phase transitions in more complex systems sharing the same basic structure.

## 3.1 Softmax regression

The softmax regression is a supervised machine learning tecnique which is used to classify data into defined classes. Let's suppose we want to classify the data into $M$ different classes, each of them labeled by a vector $\mathbf{y}$ of the standard basis of $\mathbb{R}^M$, whose only non zero component is the $m$-th (where $m$ is the class index), namely $\mathbf{y} = (1, 0, 0...)$ for class 1; $\mathbf{y} = (0, 1, 0, ...)$ for class 2, etc. A softmax regression with $M = 2$ classes is often called logistic regression.

Let's now define a prior probability that a data vector $\mathbf{x}_i^T$ belongs to the $m'$-th class as:

$$P(y_{i,m'} = 1|\mathbf{x}_i^T, \mathbf{w}) = \frac{e^{-\tilde{\mathbf{x}}_i^T \cdot \mathbf{w}_{m'}}}{\sum_{m=1}^{M} e^{-\tilde{\mathbf{x}}_i^T \cdot \mathbf{w}_m}} \equiv P_{i,m'}(\mathbf{w}) \tag{3.1}$$

where $\tilde{\mathbf{x}}_i^T = (1, \mathbf{x}_i^T)$ and $\mathbf{w}_m$ are parameters to be found with an optimization algorithm. This is the so called softmax function and the first components of every $\mathbf{w}_m$ are called biases.

Assuming statistical independence of the data, the posterior conditional probability of finding the observed data given the parameters $\{\mathbf{w}_m\}_{m=1}^M$ (likelihood) is

$$P(\mathbf{X}|\{\mathbf{w}_m\}_{m=1}^M) = \prod_{i=1}^{N} \prod_{m=1}^{M} P_{i,m}(\mathbf{w})^{y_{i,m}} \left(1 - P_{i,m}(\mathbf{w})\right)^{1-y_{i,m}} \tag{3.2}$$

Hence the cost function that we should minimize to find the optimal parameters is the *cross entropy*

$$C(\mathbf{w}) = -\log P(\mathbf{X}|\{\mathbf{w}_m\}_{m=1}^M) \tag{3.3}$$

which is the log-likelihood with opposite sign, more explicitly

$$C(\mathbf{w}) = -\sum_{i=1}^{N} \sum_{m=1}^{M} \left[y_{i,m} \log P_{i,m}(\mathbf{w}) + (1 - y_{i,m}) \log \left(1 - P_{i,m}(\mathbf{w})\right)\right] \tag{3.4}$$

It is important to stress that penalization terms can be added in softmax regression as well as in linear regression, and that the hyperparameter can be estimated through a validation process, as shown in the previous chapter.

### 3.1.1 Softmax regression for the Ising model

As an example, we performed a softmax regression on the two dimensional Ising model. In this case the task is to perform a binary classification of the phase (paramagnetic or ferromagnetic) of unknown samples; this means that the vector $\mathbf{y}$ can be either $(1, 0)$ or $(0, 1)$. In this case the second component of $\mathbf{y}$ is redundant, since the first component already identifies the classes uniquely. The classifier is then just a number 0 or 1 corresponding to the first component of the vector $\mathbf{y}$. First we generated a training dataset at eight different temperatures (half below and half above the critical value), with 1000 samples per temperature and we labeled each sample with 1 or 0 if the corresponding temperature was below or above $T_c$ respectively. All ordered states are magnetized upwards. After that we generated a test dataset at the same temperatures with 500 samples per temperature and stored the corresponding classifier $y_i^{true}$. Using the previously trained classifier function, we computed the probability $P(y_i = 1|\mathbf{x^T}_i, \mathbf{w})$, and assigned the predicted value $y_i^{pred.} = 1$ whenever it was higher than 50% and $y_i^{pred.} = 0$ otherwise.

In order to study the performance of the logistic regression, we computed the classification error at every temperature (i.e. the fraction of uncorrectly assigned samples)

$$\varepsilon = \frac{1}{N} \sum_{i=1}^{N} |y_i^{pred.} - y_i^{true}| \tag{3.5}$$

with $N = 1000$ in our example. In Tab. 3.1 we show the results of this study.

| $T/J$ | 1.2 | 1.5 | 1.8 | 2.1 | 2.4 | 2.7 | 3.0 | 3.3 |
|---|---|---|---|---|---|---|---|---|
| $\varepsilon$ | 0% | 0% | 0% | 0% | 19.6% | 6.2% | 6.8% | 7.6% |

Table 3.1: Performance of a 2-classes softmax regression in classifying unknown samples of the two dimensional Ising model ($L = 40$). Training and test datasets have been sampled at the same temperatures, but in principle one can use different temperatures. The ordered states of the training set are all magnetized upwards.

Looking at the learned parameters and biases we don't notice any relevant pattern; the only remarkable property is that the weights and biases of the two channels are opposite $\mathbf{w}_1 \approx -\mathbf{w}_2$ and $b_1 \approx -b_2$. This is a consequence of the fact that outputs of softmax function have to sum up to 1 since they are probabilities.

The softmax regression however has some limits that are worth mentioning. Tab. 3.1 has been obtained training the softmax classifier with only upwards magnetized ordered states. When training the classifier with a different dataset with ordered states of both magnetizations, the performance is much worse, as reported in Tab. 3.2. In this case we have sampled 1000 configurations per temperature, with 500 magnetized upwards and 500 magnetized downwards ordered states at every $T < T_c$.

| $T/J$ | 1.2 | 1.5 | 1.8 | 2.1 | 2.4 | 2.7 | 3.0 | 3.3 |
|---|---|---|---|---|---|---|---|---|
| $\varepsilon$ | 0% | 0.2% | 3.8% | 4.6% | 55.8% | 53% | 25.2% | 28.2% |

Table 3.2: Performance of a 2-classes softmax regression in classifying unknown samples of the two dimensional Ising model ($L = 40$). Here the training set is composed by ordered states of both magnetization.

This problem could be overcome by performing a classification into three classes instead of two (ordered spin-up, ordered spin-down and disordered), but in this work we keep two classes and develop more sophisticated tools to improve the classification.

## 3.2 Feed-forward neural networks

The techniques that we have discussed so far are basic examples of more general constructs called feed-forward neural networks. The basic element of a network is called *neuron* for analogy to human brain (or *unit*), and neurons are organized in layers. Each neuron in a layer is connected to all neurons in the previous and following layers (in this case the layer is said to be fully connected, but in principle a single unit could be linked to few other units). Each neuron stores a number according to some rules which will be clear later.

Layers are divided in three cathegories: input, hidden and output layers. An input layer is an array of neurons filled with an input dataset; output layers are neurons carrying the numeric result of the procedure encoded in the network; hidden layers are made by internal neurons used to perform partial operations. The array stored in a the $i$-th layer with $n_i$ neurons ($\mathbf{x}_i$) depends on the numbers in the previous layer

$$\mathbf{x}_i = f\left(\mathbf{w}_i \mathbf{x}_{i-1} + \mathbf{b}_i\right) \tag{3.6}$$

where $\mathbf{w}_i$ is a $n_i \times n_{i-1}$ matrix of parameters called weights and $\mathbf{b}_i$ is a vector of $n_i$ parameters called biases. The activation function $f$ is suitably choosen when building the network's architecture, while the parameters are determined after a training process minimizing a cost function $C(\mathbf{w}_1, \mathbf{b}_1, ..., \mathbf{w}_k, \mathbf{b}_k)$, where $k$ is the number of non-input layers. In this language $\mathbf{x}_0$ is the input $n_0$-dimensional vector of data, while $\mathbf{x}_k$ is the output $n_k$-dimensional vector of data. A schematic picture of a simple neural network with three layers is represented in Fig. 3.1.

It is worth remarking that here we have described feed-forward networks, where an input is passed to the network, which performs consecutive evalutaions and in the end returns an output. In other words, each layer determines the outcome of the following one. For some purposes though, more sophisticated network structures are used, a famous example being Restricted Boltzmann Machines (Ref. [7]).

Linear and softmax regressions are the simplest examples of neural networks. Their simple structure is due to the lack of hidden layers, indeed they only have an input layer followed by the output layer, and $k = 1$. In the case of linear regression the output layer is made by one neuron, the activation function $f$ is the identity, the bias vector is set to 0 and the cost function is given by Eq. 2.4 or Eq. 2.12 depending on the form of the regularization term. Such a simple network is known in literature as *Perceptron*. In the softmax regression for data classification into $M$ classes the output layer is made by $M$ neurons, the matrix of weights has $M$ rows, the bias vector has $M$ components and the activation function is the so called softmax function introduced in Eq. 3.1 and repeated here with suitable notation

$$\mathbf{x}_{1,m'} = \frac{e^{-\mathbf{w}_{1,m'} \cdot \mathbf{x}_0 - b_{m'}}}{\sum_{m=1}^{M} e^{-\mathbf{w}_{1,m} \cdot \mathbf{x}_0 - b_m}} \tag{3.7}$$

The cost function in this case is the cross entropy defined in Eq. 3.4, sometimes slightly modified with regularizers.

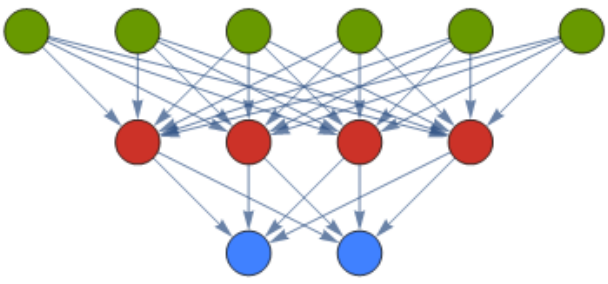


Figure 3.1: Architecture of a fully connected neural network with an input layer of 6 neurons (green), a hidden layer of 4 neurons (red) and an output layer of 2 neurons (blue). Each arrow represents a weight value (biases are not represented). Weights are organized in two matrices $\mathbf{w}_1$ (size $4 \times 6$) and $\mathbf{w}_2$ (size $2 \times 4$).

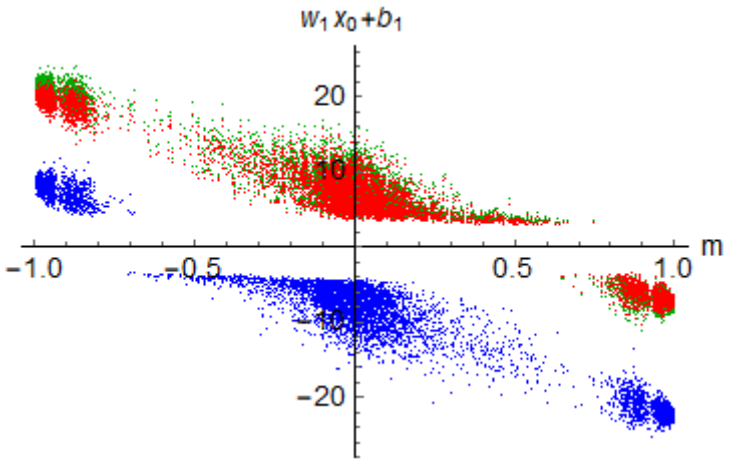



Figure 3.2: Values stored in the hidden layer before activation as a function of magnetization of the input sample. Different colors are referred to different neurons

### 3.2.1 Deep neural network for the Ising model

Whenever a neural network has hidden layers it is called *deep* (DNN). The purpose of this section is to illustrate how a deep network learns features of a system, with the explicit example of the two dimensional Ising model. In particular, we repeat the analysis and summarise the ideas of Ref. [34].

The toy model DNN architecture is the following. An input layer with 1600 neurons is used to store Ising spin configurations on a $40 \times 40$ square lattice; then a fully connected hidden layer with three neurons activated by the function $f(x) = (1 - e^{-x})^{-1}$ is connected to a softmax output layer with two units that classify ordered from disordered phases. A training set composed by 1000 disordered and ordered configurations (with both positive and negative magnetization) per temperature is sampled with the Metropolis Hastings algorithm and the network is trained. After checking the performance of the network on a test set (Tab. 3.3), we study the values stored in the three hidden neurons (Fig. 3.2).

| $T/J$ | 1.2 | 1.5 | 1.8 | 2.1 | 2.4 | 2.7 | 3.0 | 3.3 |
|---|---|---|---|---|---|---|---|---|
| $\varepsilon$ | 0% | 0% | 0% | 0% | 0% | 0% | 0.4% | 0.2% |

Table 3.3: Performance of a simple DNN with one hidden layer with three units in classifying unknown samples of the two dimensional Ising model ($L = 40$). Training and test datasets have been sampled at the same temperatures, but in principle one can use different temperatures.

Looking at Tab. 3.3 we see a surprisingly good capability of the network of classifying phases, which is much better than a simple sofmtax classifier (Tab. 3.2). The good results achieved with the deep network and shown in Tab. 3.3 must be considered carefully. Since we have sampled the test set at the same temperatures as the training set, it is possible that training and test samples have many similarities. To investigate the performance of the deep network we used the same training set and considered a different test set, with data sampled at different temperatures, covering a wider range of values (Tab. 3.4). The table shows that the deep network succeeds in classifying configurations that are completely different from the training ones. The only exception is at $T/J = 2.3$, which is the critical temperature, where the wrongly classified samples are 69% (here we assumed that samples at this temperature are disordered, but due to finite size of the system they could in principle be ordered, in this case $\varepsilon = 31\%$).

| $T/J$ | 0.5 | 0.8 | 1.1 | 1.4 | 1.7 | 2.0 | 2.3 | 2.6 | 2.9 | 3.2 | 3.5 | 3.8 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\varepsilon$ | 0% | 0% | 0% | 0% | 0% | 0% | 69% | 4.6% | 2.6% | 3.2% | 3.4% | 4.8% |

Table 3.4: Performance of a simple DNN with one hidden layer with three units in classifying unknown samples of the two dimensional Ising model ($L = 40$).

We can have a good insight on how the DNN works by looking at Fig. 3.2, where we present the values stored in the hidden neurons before activation, namely $\mathbf{w}_1\mathbf{x}_0 + \mathbf{b}_1 \equiv \mathbf{y}_1$. Let's first recall that the chosen activation function is basically a smoothened step function, since $f(x) \to 0$ when $x \to -\infty$ and $f(x) \to 1$ when $x \to +\infty$, and that a neuron is said to be activated (unactivated) when after applying $f$ it is $\approx 1$ ($\approx 0$). So when a point is in the upper half plane of the figure, the neuron will be activated, while it will be unactivated when it is in the lower half plane. As we can see from the figure, whenever the absolute value of the input sample's magnetization exceeds a quantity $|m(\mathbf{x}_0)| > m_0 \approx 0.7$, the three hidden neurons are all activated (negative magnetization) or unactivated (positive magnetization). In the opposite case, when $|m(\mathbf{x}_0)| < m_0$, only two of the three neurons are activated and the other one is unactivated.

The possible outcomes of the first hidden layer are then $\mathbf{x}_1 \approx (1,1,1)$ or $\mathbf{x}_1 \approx (0,0,0)$ (ordered phases), or $\mathbf{x}_1 \approx (0,1,1)$ (disordered phase) according to the number of activated neurons. The output layer performs a 2-classes softmax classification of the hidden layer, namely an application of the logistic function $f(x)$ to the linear combination $\mathbf{w}_2\mathbf{x}_1 + \mathbf{b}_2 \equiv \mathbf{y}_2$. The learned weights and biases are

$$\mathbf{w}_2 = \begin{pmatrix} -15.1 & 6.8 & 7.3 \\ 13.3 & -7.4 & -6.9 \end{pmatrix} \qquad \mathbf{b}_2 = \begin{pmatrix} -7.0 \\ 7.0 \end{pmatrix}.$$

Using these parameters, the hidden neurons are mapped in the output ones as follows: $(1,1,1) \to (0,1)$; $(0,0,0) \to (0,1)$ and $(0,1,1) \to (1,0)$, hence the ordered states are classified by $(0,1)$ independently from the sign of magnetization, while the disordered states are classified by $(1,0)$. In conclusion, the deep network learns the optimal threshold for the absolute magnetization: samples with magnetization above and below the threshold are distinguished.

The great research advantage provided by DNNs is the capability of generalizing beyond the training dataset, not only predicting features from other samples of the same physical system, but also of new physical systems. In Ref. [34] a generalization of the network presented here is built up and trained with standard Ising states on square lattices. The DNN is then applied to study new systems, such as Ising model with antiferromagnetic couplings and in different geometries.

## 3.3 Convolutional neural networks

A successful network architecture which has provided good results in image recognition is the *convolutional neural network* (CNN). In this case the input $X$ is an image, or more generally a rank 3 tensor having width, heigth and depth with $w \times h \times d$ components. In the case of an image $w = h = L$ and $L^2$ is the number of pixels used to describe the picture, while $d = 3$ is the number of RGB channels associated to every pixel.

The first step of the process is a convolution between the input tensor and a set of $K$ tensors called filters or kernels $F^\alpha$, $\alpha = 1, ..., K$ (each of them having $f \times g \times d$ components) with parameters to be determined during the training process. The convolution at a given channel labeled by $i, j$ consists in the following operation

$$Y_{ij}^\alpha = \sum_{p=0}^{w-1} \sum_{q=0}^{h-1} \sum_{r=0}^{d-1} X_{i+p,j+q,r} F_{p,q,r}^\alpha \tag{3.8}$$

Since this operation is clearly not possible for edge sites, it is of common use to pad the input layer with a suitable number of zeroes, so that the next layer (convolutional layer) is a tensor with $w \times h \times K$ components. Here we have described a process with stride equal to 1, but it's also possible to perform the convolution skipping some sites, for example choosing a stride of 2 means that we don't perform a convolution of the nearest neighbors of a previously convoluted site. In this case the output layer has lower dimensionality depending on the selected stride. It is also possible to choose different strides for different directions. The activation function of neurons in the convolutional layer is often the *ramp function* $R(x) = \max(0, x)$: $Y_{ij}^\alpha \to R(Y_{ij}^\alpha)$. In this case the convolutional layer is said to be a ReLU activated layer (where ReLU is an acronym for rectified linear unit).

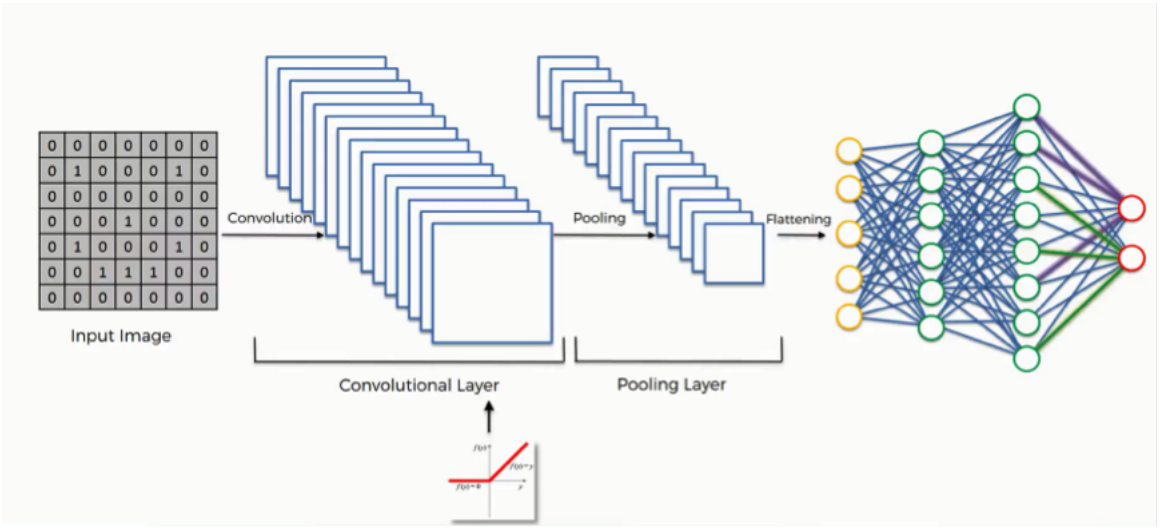



Figure 3.3: Scheme of a convolutional neural network. The input layer is organized in a $7 \times 7 \times 1$ tensor. The ReLU activated convolution with 15 filters produces a layer of depth 15 (the other dimensions depend on the choice of padding, stride and filters). A pooling procedure preserves the depth but coarse grains the previous layer. The pooling layer is flattened and used as an input for a fully connected deep neural network with 2 hidden layers, which performs a binary softmax classification.

The second step of a CNN is *pooling*, which is a coarse graining procedure that takes a convolutional layer as an input and reduces the "spatial" dimensions $w$ and $h$, while preserving depth according to some criterion. The most widespread method is called max-pooling of size $M$ and consists in mapping a $M \times M \times 1$ subtensor in its maximum value. For example, a max-pooling of size 2 is

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \rightarrow \begin{pmatrix} 6 & 8 \\ 14 & 16 \end{pmatrix}$$

This coarse graining procedure is reminiscent of the renormalization group theory used in statistical physics, and this observation has motivated many interesting studies that investigated a connection between human cognition and statistical physics; for example we mention Refs [40, 41].

Convolution and pooling can be repeated many times and eventually the resulting layer is flattened and used as an input layer for a softmax classifier (or for a deep network). After a training process, the softmax parameters and filters are optimized and the network can be used to classify unseen input data. A visual scheme of a convolutional network architecture is provided in Fig. 3.3.

### 3.3.1 Convolutional network for the Ising model

The architecture of a CNN is perfectly suited for image classification, but it can be extended to other contexts of physical interest. For example, here we show how to build a CNN for identifying different phases of the two-dimensional Ising model. In this case, an input data is organized in a $L \times L \times 1$ tensor, where $L$ is the lattice size, and depth is 1 since only one feature is associated to a lattice site ($-1$ or $+1$). For the convolutional layer we used $K = 10$ squared filters with $f = g = 2$ and 3 and depth 1 to correctly match the input. We then performed a max-pool of size $M = 2$, flattening the output layer and applying a softmax classifier. The network has been trained using the same samples we used for the deep network of the previous section and it has been applied to the same test set. In Tabs. 3.5, 3.6 we report the percentage of wrongly classified test samples $\varepsilon$ (Eq. 3.5) as a function of temperature. In the former we used a test set sampled at the same temperatures as the training set; in the latter we used different temperatures within a wider range for testing. The use of convolution and pooling steps remarkably improves the results obtained by simply applying the softmax classifier The convolutional network is successful as the DNN in classifying new data, even if quite different from the training.

| | $T/J$ | 1.2 | 1.5 | 1.8 | 2.1 | 2.4 | 2.7 | 3.0 | 3.3 |
|---|---|---|---|---|---|---|---|---|---|
| $f=2$ | $\varepsilon$ | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% |
| $f=3$ | $\varepsilon$ | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0.2% |

Table 3.5: Performance of the CNN in classifying unknown samples of the two dimensional Ising model ($L=40$). The two rows correspond to different dimensions of the chosen filters. Training and test datasets have been sampled at the same temperatures.

| | $T/J$ | 0.5 | 0.8 | 1.1 | 1.4 | 1.7 | 2.0 | 2.3 | 2.6 | 2.9 | 3.2 | 3.5 | 3.8 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $f=2$ | $\varepsilon$ | 0% | 0% | 0% | 0% | 0% | 0% | 14.8% | 0% | 0% | 0% | 0% | 0% |
| $f=3$ | $\varepsilon$ | 0% | 0% | 0% | 0% | 0% | 0% | 18.2% | 0% | 0% | 0% | 0% | 0% |

Table 3.6: Performance of the same trained CNN in classifying unknown samples of the two dimensional Ising model ($L=40$). The test set is sampled at different temperatures than the training set.

Unfortunately the working mechanism of the CNN is not as transparent as in the DNN, for which we built a simple toy model. However, an attempt of explaination is provided in Ref. [39] where the authors suggest using an ascent gradient method to find which input configuration triggers the highest value of a given neuron in the convolutional layer.

### 3.3.2 Convolutional network for the XY model

In the previous sections we have discussed how neural networks are able to learn different phases of a simple model such as the Ising model, where a simple order parameter obtained by a linear combination of input features describes the phases. As we have shown for the DNN (and guessed for the CNN), the basic idea is that the network sets an optimal threshold magnetization during the training process, and then classifies test samples checking whether or not their magnetization exceeds the threshold.

The long term perspective of research is to apply neural networks to more complicated systems, where a simple order parameter is not known, or it is given by a non linear function of the input features. A good starting point is the XY model, since the phase transition is driven by unbinding of topological defects above a critical temperature, while the magnetization vanishes in the thermodynamic limit for every temperature. In particular, we begin discussing how to build an exact CNN for classifying topological phases from configurations of angles $\{\theta_{ij}\}$. Secondly, we keep the same network structure but we let the parameters relax to a minimum of the cost function with a training process and compare the optimized network with the previous one.

The proposed network transforms the raw spin configurations into vorticity configurations $\{\theta_{ij}\} \to \{q_{ij}\}$, and then it classifies the transformed samples. The local vorticities are computed after a convolution of the input with four $2 \times 2$ kernels

$$F^1 = \begin{pmatrix} -1 & 1 \\ 0 & 0 \end{pmatrix},\ F^2 = \begin{pmatrix} 0 & -1 \\ 0 & 1 \end{pmatrix},\ F^3 = \begin{pmatrix} 0 & 0 \\ 1 & -1 \end{pmatrix},\ F^4 = \begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix}. \tag{3.9}$$

The convolution layer is then activated by the sawtooth function $f(x)$ defined in Eq. 1.36. With suitable padding and stride 1 this layer has $L \times L \times 4$ units $Y_{ij}^{\alpha}$ each one carrying the angle difference between neighbor spins rescaled in the interval $[-\pi, +\pi]$. Summing up the four units (i.e. performing another convolution with a kernel $K = (1,1,1,1)$) at fixed $i, j$, one gets the vorticity $\sum_{\alpha=1}^{4} Y_{ij}^{\alpha} = q_{ij}$. The following step is to build a classifier for the vorticity dataset. An efficient CNN architecture for this purpose has been proposed in Ref. [32]. The architecture is made by two ReLU activated convolution layers, the first one obtained with eight $3 \times 3$ filters, and the second one with sixteen $3 \times 3$ filters. These layers are then followed by a max-pool layer of size 2; the output is flattened and passed to a ReLU activated linear layer with 32 units before being classified in two classes with a softmax layer. The Wolfram code for the network architecture is reported in Fig. 18 of the Appendix.

The training process is performed in two different ways. In the first one we keep the filters of the first layer fixed (Eq. 3.9) and optimize the other parameters; in the second one we fully train the network.

| | $T/J$ | 0.3 | 0.5 | 0.7 | 0.9 | 1.1 | 1.3 | 1.5 |
|---|---|---|---|---|---|---|---|---|
| Partially trained | $\varepsilon$ | 0.5% | 0% | 0.1% | 39.4% | 0% | 0% | 0% |
| Fully trained | $\varepsilon$ | 8.7% | 5.7% | 10.9% | 28.1% | 11.1% | 0.6% | 0% |

Table 3.7: Performance of the CNN in classifying unknown samples of the two dimensional XY model ($L = 32$). The first row shows the pefrormance of the network with fixed weights in the first convolution layer (Eq. 3.9). The second row shows the performance of the fully trained network. The training dataset is made by 1000 samples per temperature at $T/J = 0.4, 0.6, 0.8, 1.0, 1.2, 1.4$.

The performance of the two networks is reported in Tab. 3.7, where we show the percentage of wrongly classified test samples. Not surprisingly we see an increasing error in classifying test samples at the critical point $T_{KT}/J \approx 0.9$, while at the other temperatures the accuracy increases. What is actually surprising is that the two networks behave differently at every temperature, due to the fact that the fully trained network learns different weights for the first two convolutional layers than the partially trained one. The learnt filters after a full training process are:

$$\begin{pmatrix} 0.86 & 0.40 \\ -0.34 & 0.43 \end{pmatrix}, \begin{pmatrix} 0.88 & 0.85 \\ 0.58 & -0.71 \end{pmatrix}, \begin{pmatrix} 0.09 & 0.37 \\ 0.59 & -0.90 \end{pmatrix}, \begin{pmatrix} 0.72 & 0.14 \\ -0.04 & 0.57 \end{pmatrix}, \tag{3.10}$$

$$\begin{pmatrix} 0.21 & 0.76 & -0.92 & 0.34 \end{pmatrix}. \tag{3.11}$$

The conclusion of this analysis is that a freely trained convolutional network with the above structure does not convert spin configurations to vorticity configurations to classify phases of XY model. Moreover, even if forced to use vorticity configurations, it is not clear what features the CNN tries to extract from the input (density of topological defects, distance between vortices, etc.). A detailed study on this topic is provided in Ref. [32], where the authors performed the same analysis for different lattice sizes. Even if they could not conclude what the network was learning exactly, they proved that the freely trained network outperforms the engineered network for $L \ll 32$, while the opposite happens for $L \gg 32$. They also concluded that the parameters of the engineered network are a stable local minimum for the cost function, since a network initialized at those parameters and then freely trained does not change those parameters.

# Chapter 4

# Detecting phase transitions with unsupervised learning

In this chapter we discuss the physical problem of detecting phase transitions with machine learning. The long term perspective of this work is to detect phase transitions in complex systems where the order parameter and the critical properties are unknown, so we use the tools of unsupervised machine learning. Unsupervised learning consists in extracting information from the dataset without previous training of the algorithm. The basic concepts behind unsupervised learning are dimensional reduction and clustering. The former means projecting data into a space with a dimension lower than the one of the input data, trying to keep track of the most relevant patterns. The latter means dividing data into clusters according to some criterion of similarity, in order to understand properties of the input. Since unsupervised learning does not require any prior knowledge of the system from which data are extracted, these are the most interesting techniques to implement for our purpose. As a first step, we apply these techniques to simple spin models (Ising, Potts and XY), whose critical properties are well-known from Statistical Mechanics, so that we can compare our results to exact solutions.

## 4.1 Principal Component Analysis

So far we have seen the main methods of supervised learning: a training data set with known properties is used to optimize some parameters, which are then used to predict properties of unknown data (test). To apply these methods we need some prior knowledge about the data, but this is not always possible. In some cases we don't know anything about our data and need to extract some information from them. For this purpose we have to implement *unsupervised learning* algorithms.

One of the simplest and most used methods is the *Principal Component Analysis (PCA)*, which we present in this section. This method is based on the diagonalization of the correlation matrix of the input data

$$\mathbf{\Sigma} = \frac{1}{N-1}\mathbf{X}^T\mathbf{X}, \tag{4.1}$$

where $\mathbf{X}$ is the design matrix, which is a rectangular matrix with $N$ rows (number of samples) and $p$ columns (number of features) containing the dataset. A full exact diagonalization of $\mathbf{\Sigma}$ can be computationally tough, especially when $p$ is great; and when also $N$ is large it is even challenging to compute the matrix product. Since in practice one does not need a full diagonalization, but just the highest $\bar{p}$ eigenvalues, one can use a technique of linear algebra called *Singular Value Decomposition* (SVD) to reduce the computational cost.

Performing a SVD of the design matrix means finding three matrices $\mathbf{U}$, $\mathbf{S}$ and $\mathbf{V}$ such that

$$\mathbf{X} = \mathbf{USV}^T, \tag{4.2}$$

where $\mathbf{U}$ and $\mathbf{V}$ are $N \times N$ and $p \times p$ unitary square matrices respectively ($\mathbf{UU}^T = \mathbb{1}$, $\mathbf{VV}^T = \mathbb{1}$) and $\mathbf{S}$ is a $N \times p$ rectangular matrix with the only non zero terms, called singular values, being on

the main diagonal ($\mathbf{S}_{ij} = 0\ \forall i, j$ with $i \neq j$). The SVD of any rectangular matrix with real entries is guaranteed to exist, and its singular values are unique. For demonstrations, algebraic and geometrical background and applications we reccomend Refs. [24, 25].

Using the new matrices it is possible to diagonalize the covariance matrix

$$\mathbf{\Sigma} = \frac{1}{N-1}\mathbf{X}^T\mathbf{X} = \mathbf{V}\left(\frac{\mathbf{S}^T\mathbf{S}}{N-1}\right)\mathbf{V}^T \equiv \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T, \tag{4.3}$$

where each row of $\mathbf{X}$ is supposed to have zero mean, without loss of generality. $\mathbf{\Lambda}$ is a $p \times p$ diagonal matrix and its elements $\mathbf{\Lambda}_{ii} \equiv \lambda_i$ are the eigenvalues of the covariance matrix. Multiplying both sides of Eq. 4.3 by $\mathbf{V}$ to the right, we get $\mathbf{\Sigma}\mathbf{V} = \mathbf{V}\mathbf{\Lambda}$, which means that the column vectors of $\mathbf{V}$ are the eigenvectors of $\mathbf{\Sigma}$.

An interesting parameter to compute once we have performed the SVD is the ratio $\lambda_i / \sum_{j=1}^{p} \lambda_j$, which is called *explained variance* of the $i$-th feature and is an index of data variation along the direction of the $i$-th eigenvector ($i$-th column of $\mathbf{V}$). In many contexts it happens that the main variability of the data is explained by the first $\bar{p}$ features, with $\bar{p} \ll p$, thus suggesting that the intrinsic dimensionality of the data is much smaller than $p$. We can thus project our dataset in the subspace of dimension $\bar{p}$ defined by the first $\bar{p}$ principal vectors building a $p \times \bar{p}$ matrix $\tilde{\mathbf{V}}$ made by the first $\bar{p}$ columns of $\mathbf{V}$ and then computing $\mathbf{Y} = \mathbf{X}\tilde{\mathbf{V}}$. $\mathbf{Y}$ is now a $N \times \bar{p}$ matrix which represents $N$ samples of our data in a $\bar{p}$ dimensional space; the total explained variance with this approximation is simply $\sum_{i=1}^{\bar{p}} \lambda_i / \sum_{j=1}^{p} \lambda_j$. Every row of $\mathbf{Y}$ represents a point in $\mathbb{R}^{\bar{p}}$, which is the space of the principal components, so that at the end of this procedure we get a set of $N$ points in $\mathbb{R}^{\bar{p}}$. Plotting these points we can get some insight into the physical problem.

### 4.1.1 PCA for the Ising model

In the case of an Ising model in a two dimensional square lattice of size $L = 40$, we can see that the first two principal components explain the 99.9924% of the total variance, as shown in Fig. 4.2, where we show the explained variance of the first 8 principal components. In Fig. 4.3 we present the projection of the data in the subspace of the first two principal directions (the dataset $\mathbf{Y}$). It is clear from this figure that data taken from different phases of the Ising model are grouped in well defined clusters: samples in the paramagnetic phase (i.e. extracted at $T > T_c$) are in the center of this graphic (red), while samples in the ferromagnetic phase (extracted at $T < T_c$) are in the two extrema (magenta), each one corresponding to a phase with different magnetization (in the graphic we have plotted only samples with positive magnetization, which are grouped to the left, in order to remark the spontaneous symmetry breaking).

The PCA also provides a more quantitative result for phase recognition because we can use it to build an order parameter without any prior knowledge on the system. Looking at Fig. 4.2 and 4.3, we can convince ourselves that the first principal component is the most relevant for phase recognition, so we can plot the first column of the matrix $\tilde{\mathbf{V}}$ as a function of two indices $i, j$ labeling the lattice sites. The result is shown in the upper left panel of Fig. 20 (Appendix), from which we see that all the sites are combined with the same weight $v$. A good order parameter is then

$$y_1 = \mathbf{x}^T \cdot \mathbf{v}_1 \approx v \sum_{i=1}^{L}\sum_{j=1}^{L} S_{ij}, \tag{4.4}$$

which can be mapped into the standard magnetization $m = \frac{1}{L^2}\sum_{ij} S_{ij}$ simply by rescaling $y_1 \to y_1/(vL^2) = m$. Since $v \approx -1/L$ (this result is true for all the sizes we have used in the following, i.e. $L = 20, 30, 40, 60$), the mapping is actually $y_1 \to -y_1/L$. In Fig. 4.1 we present the average of the modulus of $|y_1|/L$ at each temperature compared to the average modulus of the magnetization per site obtained by the Monte Carlo sampling and we show the deep connection between these two quantities.

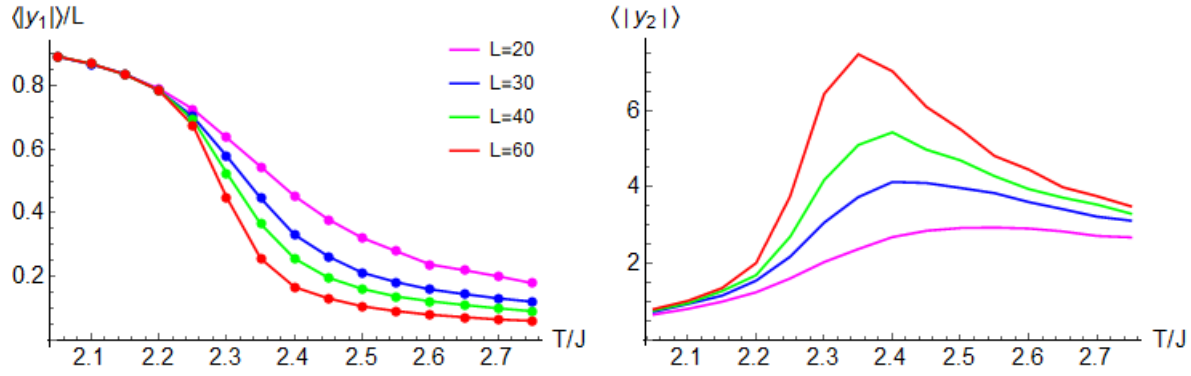



Figure 4.1: The left panel shows the average value of $|y_1|/L$ at every temperature as a function of the temperature itself for different lattice sizes (joined solid lines are used instead of points for graphical reasons). Dots are referred to the expectation value of the modulus of the magnetization per site obtained via the Monte Carlo sampling of the dataset. The right panel shows the expectation value of the modulus of the second principal component at every temperature for various lattice sizes. The color scheme is the same as in the upper panel. The peaks of this figure are more pronounced for larger systems and can be used to identify the critical temperature.

The second column of the matrix $\tilde{\mathbf{V}}$ is plotted in Fig. 4.4a as a function of the lattice site's position $(i, j)$ using a color scheme. In Ref. [27], the authors noticed a similarity with the function

$$\omega_{ij} = \mathbf{v}_2(i + (j-1)L) = \frac{1}{L}\left[\cos\frac{2\pi i}{L} + \cos\frac{2\pi j}{L}\right] \tag{4.5}$$

plotted in Fig. 4.4b for comparison. The second principal component is given by

$$y_2 = \mathbf{x}^T \cdot \mathbf{v}_2 = \sum_{i=1}^{L}\sum_{j=1}^{L} \omega_{ij} S_{ij}. \tag{4.6}$$

To have a deeper insight into the meaning of $y_2$, we can imagine how its absolute value behaves for ordered, disordered or critical samples. If the input sample is completeley ordered, say $S_i = +1\ \forall i$, then $y_2 = \sum_{ij}\omega_{ij} \approx 0$, since the weights have vanishing average. If the input is completely disordered, then the correlation lenght is small $\xi \approx 0$, and the weights are combined with opposite signs without a precise scheme; so again we expect the result to be zero. Finally, if the input sample is critical, then $\xi \approx L$, so a configuration with all spins up in the red region and all spins down in the blue one of Fig. 4.4 is more likely to occur. When computing $y_2$ for such a configuration, one has to sum up all the positive weights and then add the negative weights again with positive sign, obtaining a non zero result. So, when studying the behaviour of the average value $\langle|y_2|\rangle$, we expect to observe a peak at $T \approx T_c$, which is shown in the right panel of Fig. 4.1.

An interesting conclusion is that a spin configuration triggers a high value of a principal component whenever spins are organized as the pattern of the corresponding weights' signs (i.e. components of the corresponding eigenvector of $\mathbf{\Sigma}$). Looking at the pattern of weights it is possible to understand which spin configurations PCA tries to identify to explain the variability of the input dataset. Motivated by this observation, we performed a PCA up to the 16-th principal component, and we reported the resulting weights in the Appendix (Fig. 20). The result of this investigation is that principal components are related to the most ordered structures of the input configuration, with a progressively increasing number of domain walls [27]. For a study of a system where many principal components are needed to detect the phase transition (Ising model with conserved order parameter), we recommend Ref. [26].

Once we have recognized that the system undergoes a phase transition at some temperature between $2J$ and $3J$, we can also try to estimate the value of the critical temperature $T_c$. To this scope we studied the behaviour of the principal components when varying the lattice size, in particular we generated samples in square lattices of size $L = 20, 30, 40, 60$ in a range of temperature around the estimated critical value (between $2J$ and $3J$ in this case). The behaviour of the correlation length $\xi$

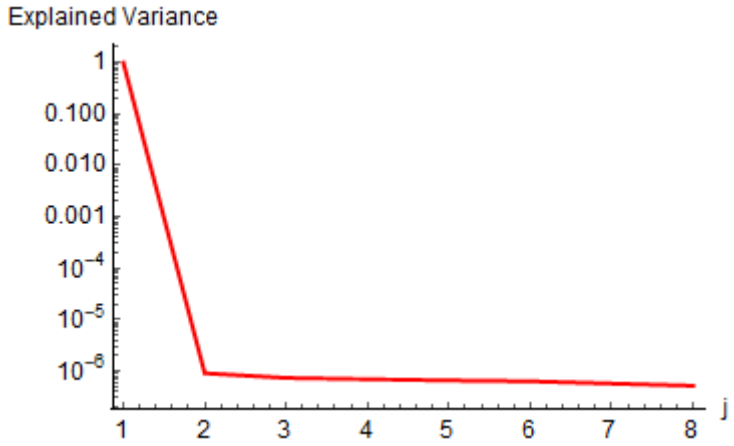



Figure 4.2: Explained variance of the first 8 principal components of the Ising model in a two dimensional $40 \times 40$ square lattice. The first principal component describes most of the variability of the data.

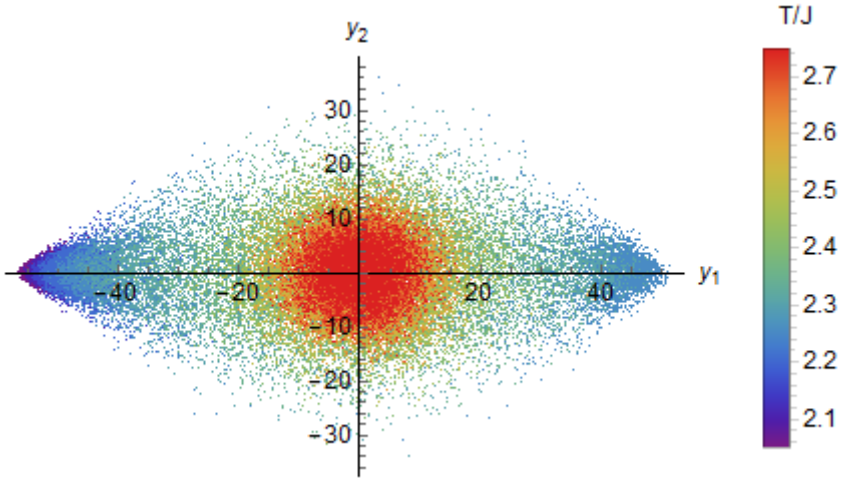



Figure 4.3: Projection of the two dimensional Ising model in a $40 \times 40$ square lattice data in the subspace of the first two principal directions. The data are generated at different temperatures, from $T/J = 2.05$ to $T/J = 2.75$, with 5000 samples for each temperature. The central red area is made of configurations in paramagnetic phase, while the magenta area to the left (the darkest part) contains configurations in the ferromagnetic phase with positive magnetization.

when the temperature is close to the critical value is given by

$$\xi \sim (T - T_c)^{-\nu} \tag{4.7}$$

where the critical exponent $\nu$ should be $\nu = 1$ according to Onsager's theory. Let's call $T_0(L)$ the temperature that maximizes the magnetic susceptibility when the lattice has size $L$. When the finite sized system has a temperature close to $T_0(L)$, the spin correlation take the size of the system and so $\xi \sim L$. Moreover, if the system has finite size, then we expect $T_0(L)$ to be close to the critical value $T_c$ and we can expect that $\xi$ has a critical behaviour. As a consequence of all this remarks we can write

$$L \sim (T_0(L) - T_c)^{-\nu} \rightarrow T_0(L) = T_c + k\frac{1}{L^\nu} \tag{4.8}$$

We can now treat $T_c$, $k$ and $\nu$ in this equation as parameters of a fit model. The interpolation is shown in Fig. 4.5 and $T_c/J = 2.25$, which is in good agreement with the theoretical value $T_c/J = 2.269$, with a 0.8% of relative error. Moreover, as expected, $\nu = 1.0$ and there is a linear relation beteween $T_0(L)$ and $1/L$.

We conclude that a studying systems of increasing sizes (finite size analysis) allows to predict critical properties such as the critical temperature $T_c$ and the universal exponent $\nu$.

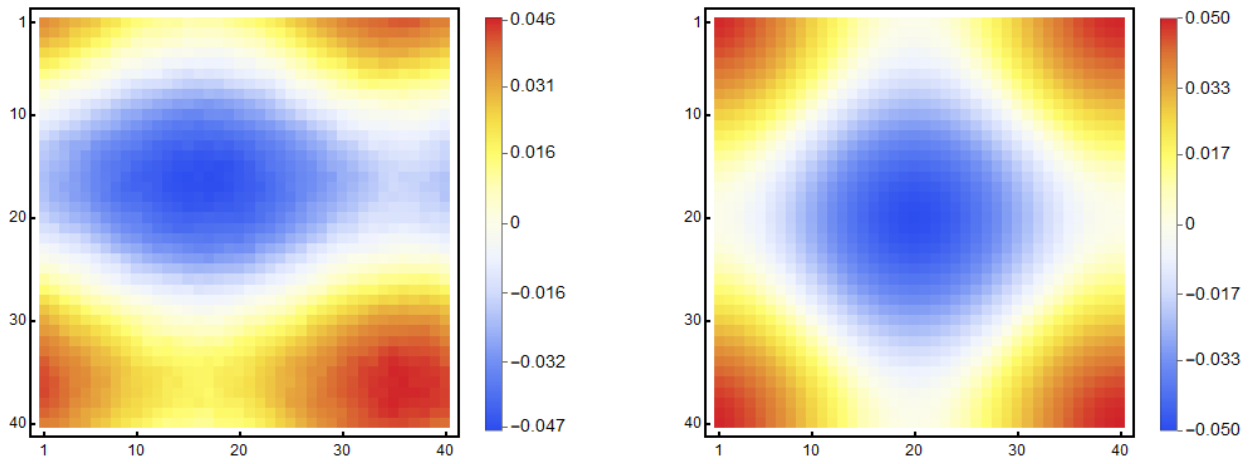



Figure 4.4: Left panel: second column of the $\tilde{\mathbf{V}}$ matrix plotted in the lattice with a color scheme. A weight is assigned to every lattice site, multiplying the local spin by this weight and taking the sum over the lattice one obtains the second principal component. Right panel: guessed function describing the pattern of weights.

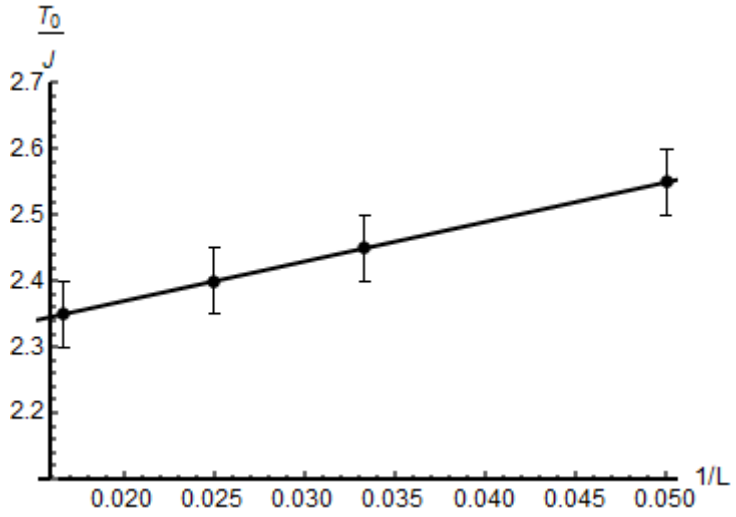



Figure 4.5: Interpolation between $1/L$ and $T_0/J$: the line has parameters $T_c/J = 2.25$, $\nu = 1.00$ and $k/J = 6.00$.

### 4.1.2 PCA for the Potts Model

After studying the Ising model with PCA, the natural following step is to analyze its behaviour on a slightly more general model, such as the $q$ state Potts model. In particular we investigate whether (and how) PCA is able to build an order parameter for a phase transition driven by a $Z_q$ symmetry breaking, and whether it allows to distinguish between first order and continuous phase transitions. Moreover we try to build the function $T_c(q)$ that gives the critical temperature as a function of the number of states. The latter task is carried out a bit naively, in order to capture the essential concepts without long running simulations, even though we will propose a more rigorous approach. The analysis is carried out as in the previous section, namely deriving the explained variance of the first principal components, then deciding a reasonable dimension for the subspace where we then project the dataset. Then we study the temperature behaviour of the first principal components and try to deduce critical properties.

First of all we perform a Monte Carlo simulation of the two dimensional model on a square lattice of size $L = 40$, at $q = 3, 9, 27, 81$ and within a temperature range containing the transition temperature for all the chosen $q$ ($0.2J < T < 1.2J$). We sample 1000 equilibrium configurations at every temperature and states number and we choose as possible states the numerical integer values in $[-(q-1)/2, +(q-1)/2]$ to obtain more symmetric graphics. For all the chosen $q$ a first principal components emerges, so we can again project data into a two dimensional linear subspace of the first two components.

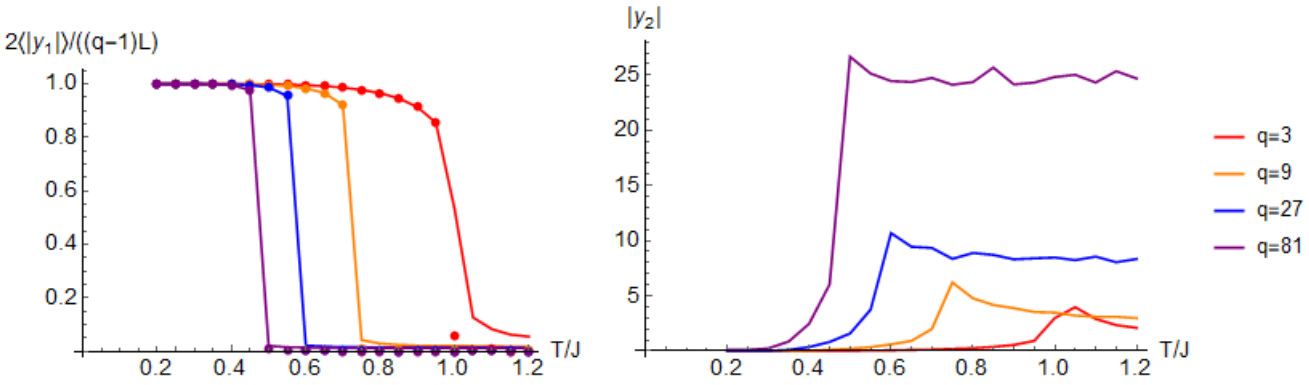



Figure 4.6: The left panel presents the expectation value of the deduced order parameter $2|y_1|/L(q-1)$ as a function of temperature (solid lines) compared to the true order parameter $s$ computed with Monte Carlo simulations (dots). The right panel presents the expectation value of $|y_2|$ at the same temperatures.

Looking at the weight vector $\mathbf{v}_1$ we understand how PCA extracts the order parameter: since every component is approximately $-1/L$, we get

$$y_1 = \mathbf{x}^T \cdot \mathbf{v}_1 = -\frac{1}{L}\sum_{ij}\sigma_{ij} \tag{4.9}$$

where $\sigma_{ij}$ is the state at site $(i,j)$. Deep in the disordered phase, a given site has the same probability of being in all the $q$ states, and since we have symmetrized the states around 0, the expectation value of $y_1$ is 0. In the ordered phase, almost all the lattice sites share the same state $\sigma$ ($\sigma = -(q-1)/2$ in our simulations), therefore $\sum_{ij}\sigma_{ij} \approx L^2\sigma$. A suitably normalized order parameter extrated from PCA is then

$$s = \frac{|y_1|}{L|\sigma|} \tag{4.10}$$

In the right panel of Fig. 4.6 we show the behaviour of the extracted order parameter with temperature. It is clear from the graphic that a phase transition occurs at some critical temperature depending on the number of states $q$. It is clear as well that the order parameter is the $s$ parameter introduced in the mean field analysis of Potts model (Chapter 1, Eq. 1.16), which is obtained by Monte Carlo simulations and shown in the figure.

The expectation value of the absolute value of the second principal component is zero in the ordered phase, then it rapidly increases at the critical temperature and finally it is almost constant (or slowly decreases) in the disordered phase. The location of the peak after the jump allow an identification of the critical temperature, so we can check the behaviour of the critical temperature with $q$ (Fig. 4.7). Since the temperature resolution of the input data is 0.05, we choose this quantity as the size of the error bar. The physical meaning of the second principal component is not clear, since we haven't argued any similarity to other physical quantities (such as specific heat or susceptibility) and the weights $\mathbf{v}_2$ do not follow a clear pattern.

One of the purposes of this study is to see whether or not PCA can give an insight in classifying the phase transition. To this scope here we have performed PCA both for a continuous transition ($q = 3$) and for first order phase transitions ($q = 9, 27, 81$). The choice $q \geq 9$ guarantees that the correlation length is much smaller than the lattice size $\xi/L \leq 0.37 \ll 1$, which is a necessary condition to observe a first order transition in a finite system. The second principal component is given by

$$y_2 = \mathbf{x}^T \cdot \mathbf{v}_2 = \sum_{ij}\omega_{ij}\sigma_{ij}, \tag{4.11}$$

where the weights $\omega_{ij}$ are the $L^2$ components of the second eigenvector of the correlation matrix organized in a square matrix of size $L$, as in the case of the Ising model. Looking at the pattern of weights, shown in Fig. 4.8, we observe that in the case of a continuous phase transition ($q = 3$) their

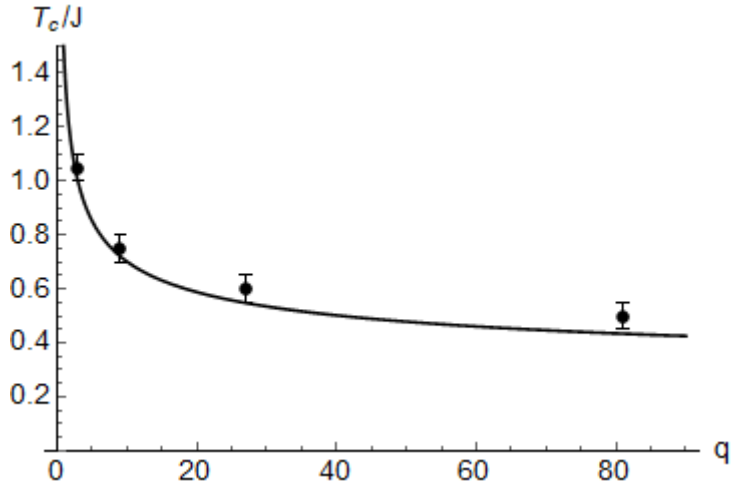



Figure 4.7: Behaviour of the critical temperature as a function of the number of states $q$ as predicted by PCA (dots), compared to the exact solution $T_c/J = 1/\log(1+\sqrt{q})$ (solid line).

distribution is similar to the one we observed for the Ising model in Fig. 4.4, and is likely to be a sum of cosines. When $q$ increases and the transition is deeply first order (with $\xi \ll L$), the pattern becomes more chaotic and eventually it appears as a random variable with gaussian distribution for $q = 27, 81$.

Even if we have found different behaviours of the weights $\omega_{ij}$ for a second and first order transition, we believe that a clear evidence that PCA is capable of distinguishing requires further investigation. It is remarkable that the phase space for the Potts model has dimension $q^{L^2}$, so the $q = 81$ model has a phase space which is $27^{L^2}$ times larger than the $q = 3$ model. It is possible that the extracted samples do not represent such a big phase space properly, so that the learnt weights are basically a random noise. On the other hand, using small values of $q$ requires large systems in order to have $\xi \ll L$, but this is computationally tough.

Let's now suggest an idea for further research on this problem. As suggested in Ref. [11], one can distinguish first order and continuous transition on the basis of a finite size scaling of physical quantities such as the Binder ratio and the critical probability density function of magnetization. One should then have access to samples of systems with different sizes $L$, which is not always possible. For instance, when the transition is *weakly* first order (still first order but with large correlation length compared to reasonable sizes for Monte Carlo simulations $L_{max} \approx 256$), one should sample lattices with really large sizes, which is computationally challenging, for example the $d = 2$, $q = 5$ Potts model has $\xi \approx 2500$, and a Monte Carlo simulation of such a big lattice is intractable. To overcome these problems, some solutions have been proposed. Again in Ref. [11] it is suggested that, at least for the $d = 2$ Potts model, signs of first order transitions can be found even using $L \ll \xi$. Another possibility is to develop neural networks capable of extending configurations of small systems to larger lattice sizes, for example in Ref. [38] a convolutional network is used to enlarge Ising configurations.

### 4.1.3 PCA for the XY Model

In this section we study the behaviour of PCA in the analysis of the more subtle phase transition of the XY model. As we have already mentioned in Chapter 1, there is no spontaneous symmetry breaking of $O(2)$ in this model, and hence a ferromagnetic phase with non vanishing magnetization does not exist at any finite temperature.

As we have seen with the Ising and Potts model, when a paramagnetic-ferromagnetic phase transition occurs, PCA identifies the magnetization as an order parameter and classifies the phases using this parameter. Since in this case the magnetization is always zero, we test the performance of PCA and look wether or not it can properly classify phases. In particular, the following analysis is performed on a standar XY model in a square lattice of size $L = 20$. Data consist in a vector $(\cos\theta_{ij}, \sin\theta_{ij})$

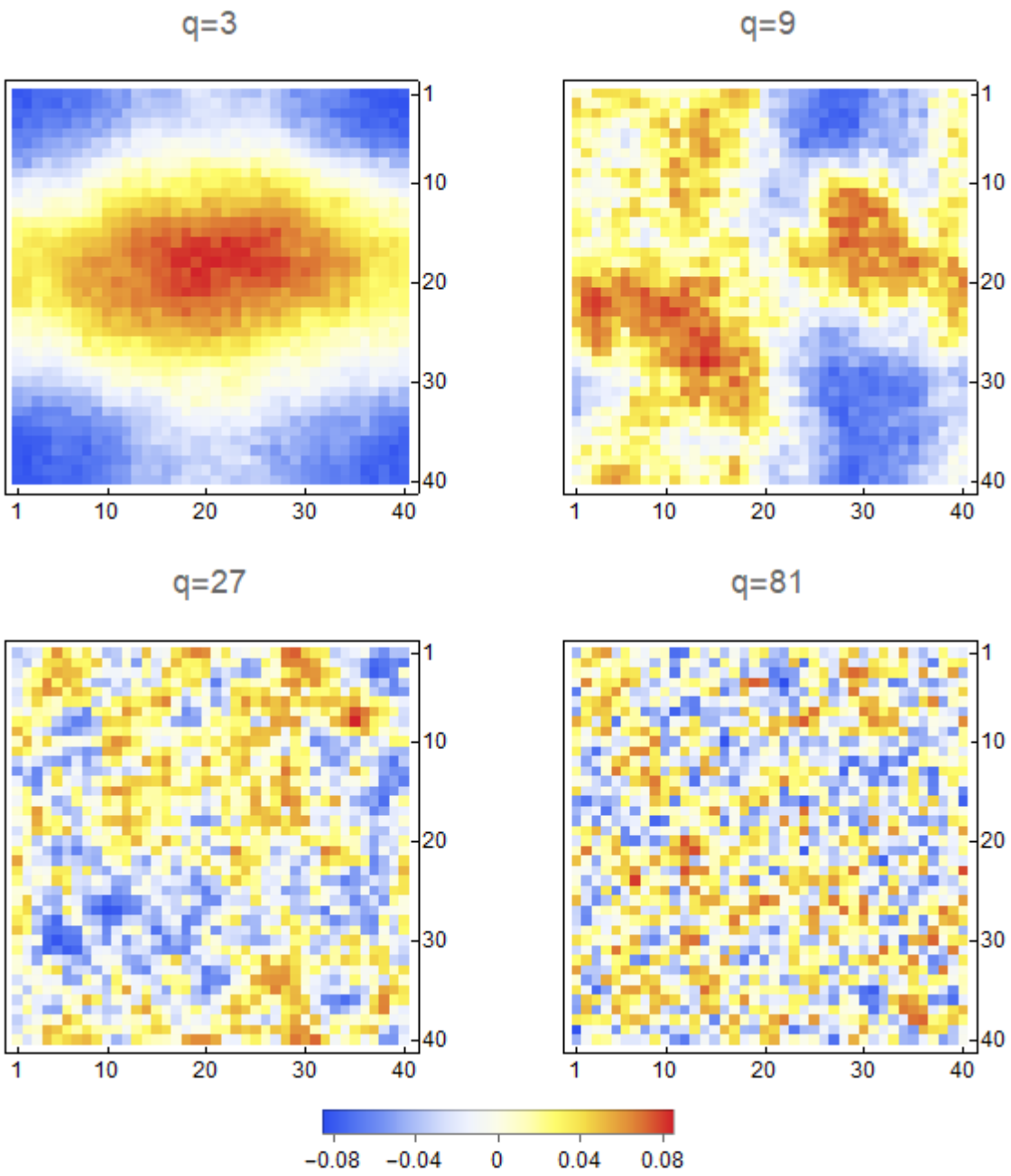



Figure 4.8: Weights for computing the second principal component of the Potts model, presented as a color scheme on the lattice. The $q = 3$ model has weights that follow a pattern similar to the one observed for the Ising model (Fig. 4.4). The pattern becomes more chaotic when $q$ increases.

associated to every lattice site of labels $i, j$, where $\theta_{ij}$ is the angle between the spin vector at that site and the $\hat{x}$ axis.

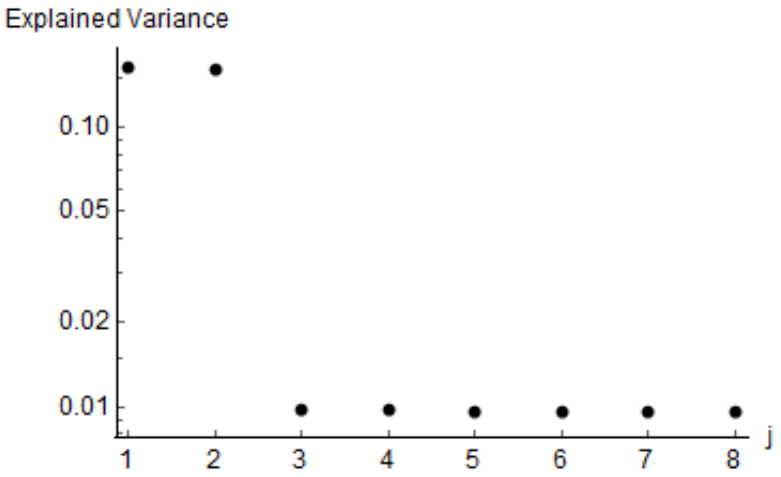



Figure 4.9: Explained variance of the first 8 principal components of the XY model. The first two components are much more relevant than all the others, so we can project the dataset in the corresponding two-dimensional subspace. The 24% of the total variance is explained by the first two components.

In Fig. 4.9 we see that the first two principal components are both equally relevant in this study, while all the others can be neglected at a first stage. With this choice we describe the 24% of the dataset's variance, but to go beyond this analysis one has to consider much more principal components, and the consequent study is more complicated.

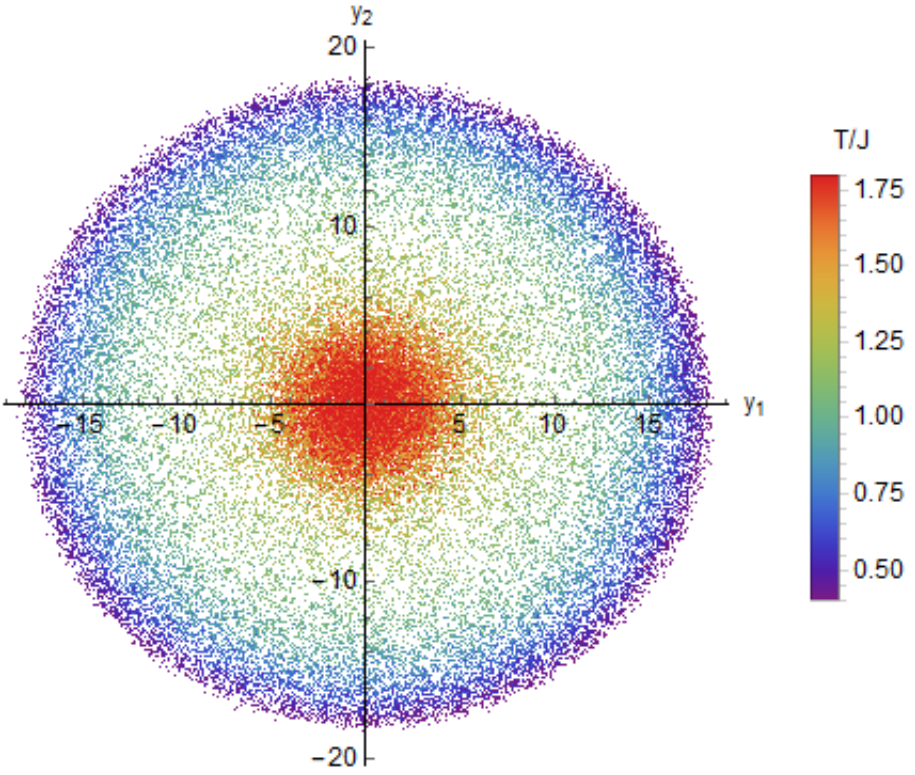



Figure 4.10: Projection of the XY spin dataset in the subspace of the first two principal components. Low temperature samples (blue) are distributed in the external circle, while high temperature samples (red) are around the origin. Critical data are distributed in the middle. A rotational symmetry clearly appears and two phases are distinguished.

In Fig. 4.10 we present the scatter plot of the XY model in the plane of the first two principal components. There is a clear rotational symmetry about the origin in the $y_1$-$y_2$ plane, which is a consequence of the $O(2)$ symmetry of the Hamiltonian. This, together with what we have shown in

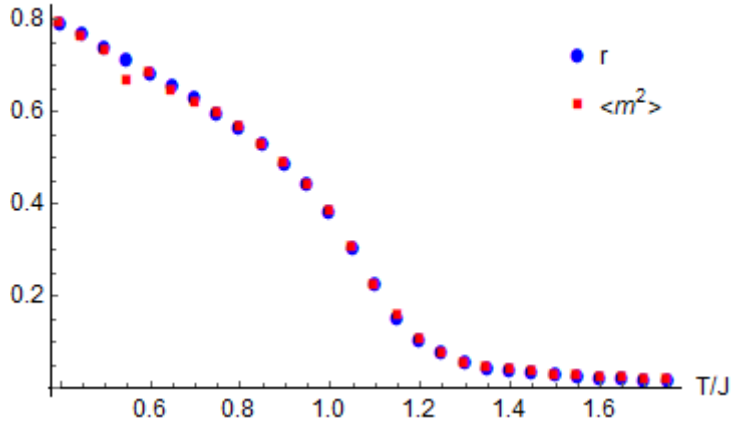



Figure 4.11: Plot of the $r$ PCA classifier as a function of the temperature. The fluctuation of the magnetization per site obtained by the Monte Carlo simulation is also plotted for comparison.

Fig. 4.9 suggests that there is no hierarchy between the first two components, and therefore plotting separately $y_1(T)$ and $y_2(T)$ is not particularly illuminating in this case. We should instead look for a combination of the two components, and the rotational symmetry immediately suggests to consider the distance from the origin $R = \sqrt{y_1^2 + y_2^2}$.

Looking at the structure of the vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ it is possible to understand the physical meaning of the classifier $R$ proposed by the PCA. In our case we notice that every component of $\mathbf{v_1}$ is approximately equal to a constant $v \approx -0.035$ (for $L = 20$), while the components of $\mathbf{v_2}$ are approximately $\pm v$ (the $+$ sign is for even components, the $-$ for the odd ones). As a consequence

$$y_1 = \mathbf{x}^T \cdot \mathbf{v_1} \approx v \sum_{ij} (\cos\theta_{ij} + \sin\theta_{ij}) = v\,(M_x + M_y) \tag{4.12}$$

$$y_2 = \mathbf{x}^T \cdot \mathbf{v_2} \approx -v \sum_{ij} (\cos\theta_{ij} - \sin\theta_{ij}) = -v\,(M_x - M_y) \tag{4.13}$$

where $M_x = \sum_{ij} \cos\theta_{ij}$ and $M_y = \sum_{ij} \sin\theta_{ij}$ are the components of the total magnetization vector. The classifier then becomes

$$R^2 = y_1^2 + y_2^2 \approx 2v^2 \left(M_x^2 + M_y^2\right) = 2v^2 L^4 \left(m_x^2 + m_y^2\right) \tag{4.14}$$

where we have introduced the magnetization per site $\vec{m} = \vec{M}/L^2$. Averaging the square radius of all the data points at the same temperature and normalizing by the number of sites $L^2$ we get

$$r \equiv \frac{\langle R^2 \rangle}{L^2} \approx 2v^2 L^2 \left\langle m_x^2 + m_y^2 \right\rangle \approx \left\langle m^2 \right\rangle \tag{4.15}$$

since the quantity $2v^2L^2 \approx 1$ not only for $L = 20$, but presumably for every $L$. Since the Mermin-Wagner theorem ensures that no spontanous symmetry breaking of $O(2)$ occurs in two dimensions, then the expectation value of the magnetization vanishes at every temperature $\langle \vec{m} \rangle = 0$ and the fluctuation of magnetization is simply given by $\left\langle m^2 \right\rangle$.

In conclusion, the classifier of the phase transition $r$ that emerges from the PCA has the physical meaning of fluctuation of the magnetization per site and is strictly connected to the magnetic susceptibility. Since this quantity is strictly related to the spatial correlation function $\Gamma(r)$ ($\chi \approx \int d^2 r \Gamma(r)$, as shown in Ref. [6]), the PCA is suggesting to look at the spatial correlations between spins to understand the phase transition. In Fig. 4.11 we plot the behaviour of $r$ as a function of the temperature compared to the magnetization fluctuations obtained by the Monte Carlo simulation.

The Kosterlitz-Thoules phase transition is a topological phase transition associated to the unbinding of vortex and antivortex pairs when increasing temperature; however PCA does not explain the topological structure of the model. One attampt to solve this problem has been performed in Ref. [27]

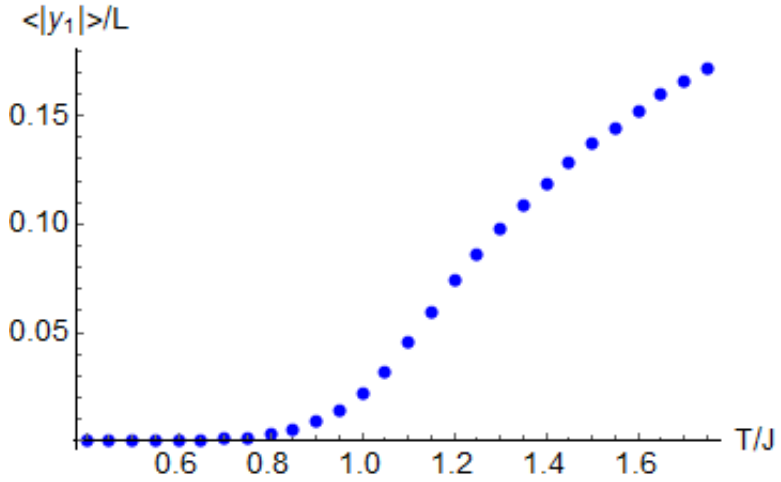



Figure 4.12: Expectation values of the first principal component rescaled by the lattice size $\langle |y_1| \rangle / L$ as a function of temperature when we feed PCA with the $\{|q_{ij}|\}$ dataset. When $T < T_{KT} = 0.89J$, less than 1% of the plaquettes contain vortices; while when $T > T_{KT}$ a proliferation of vortices is observed (about 18% of the plaquettes contain vortices at the highest sampled temperature).

by feeding PCA with a different dataset, namely dividing the square lattice in small *plaquettes* and associating to every plaquette the values $\pm 1$ or 0 depending on vorticity. This attempt however has shown the limits of PCA since all the components have about the same percentage of explained variance and it is not possible to find a subset of relevant parameters. A simple solution suggested by the authors and implemented in this work is to feed PCA with the absolute value of vorticity. With this dataset, the first principal component has an higher explained variance compared to the other components (about 83% compared to less than 2% for every other component), so we can restrict to a one dimensional subspace of data.

Feeding PCA with the set of $\{|q_{ij}|\}$ and then plotting the average value of the first principal component $|y_1|$ as a function of temperature, we can see marks of a phase transition (Fig. 4.12). Since all the entries of the vector $\mathbf{v}_1$ are approximately constant and equal to $v = -1/L$, we get

$$|y_1| = \mathbf{x}^T \cdot \mathbf{v}_1 \approx \frac{1}{L} \sum_{ij} |q_{ij}| \tag{4.16}$$

The quantity $|y_1|/L$ then physically represents the density of topological defects (both vortices and antivortices) $(1/L^2) \sum_{ij} |q_{ij}|$ of a given configuration. Looking at Fig. 4.12 we see that essentialy no vortices appear in low temperature configurations, while they become significant when the temperature is greater than the critical theoretical value $0.89J$.

## 4.2 K-means clustering

Clustering is an unsupervised machine learning task that consists in grouping data into $K$ classes according to some rules. There are many ways to cluster a given dataset, but in this section we will focus on the $K$-means clustering, which is intuitive and of relatively simple implementation.

Let $\mathbf{X}$ be the design matrix with $n$ measures (rows) and $p$ features (columns): every measure can be interpreted as a point in the configurations space $\mathbb{R}^p$. Then it is possible to introduce a notion of distance between two points in this space, for example the standard euclidean distance $\|.\|_2$. Let's suppose we have to classify data into $K$ different classes, where $K$ is known a priori and it is the only extra information required. Every cluster has a mean value $\mu_k$

$$\mu_k = \frac{\sum_{i=1}^{n} r_{ik}\mathbf{x}_i^T}{n_k} \tag{4.17}$$

where $r_{ik} = 1$ if the point labeled by $i$ belongs to the $k$-th cluster and 0 otherwise, and $n_k$ is the number of point in the $k$-th cluster. Every cluster has a *moment of inertia* $I_k$ that measures how distant the points in the cluster are from the average value

$$I_k = \sum_{i=1}^{n} r_{ik} \left\|\mathbf{x}_i^T - \mu_k\right\|_2^2 \tag{4.18}$$

The goal of $K$-means clustering is to find optimal values $\mu_k$ and $r_{ik}$ that minimize the total moment of inertia $I = \sum_{k=1}^{K} I_k$.

In practice, one starts randomly clustering the dataset (i.e. choosing $\{r_{ik}\}$ for every $i, k$), then computing the corresponding set of means $\{\mu_k\}$. At this point, one chooses all the points one by one and re-assigns that point to the cluster with the closest average value: formally speaking

$$r_{ik} = \begin{cases} 1 & k = \arg\min_{k'} \left\|\mathbf{x}_i^T - \mu_{k'}\right\|^2 \\ 0 & \text{otherwise} \end{cases} \tag{4.19}$$

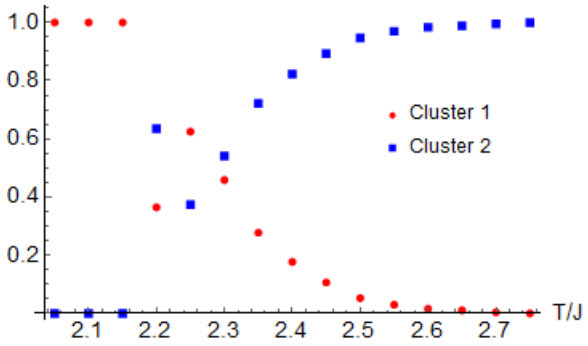
One then updates the means and repeats all the steps until a convergence rule is satisfied.

Let us briefly overview the key properties of this algorithm without giving detailed proofs. First of all, the algorithm is guaranteed to converge, but since $I$ is non convex, it could converge to a local (and non global) minimum. To overcome this problem one repeats the algorithm several times with different initialization values of $\{r_{ik}\}$ and then choses among the results the one with lowest $I$. Secondly, the algorithm requires relatively small computational effort (order $\mathbb{O}(Kn)$), since in typical situations $K \ll n$.

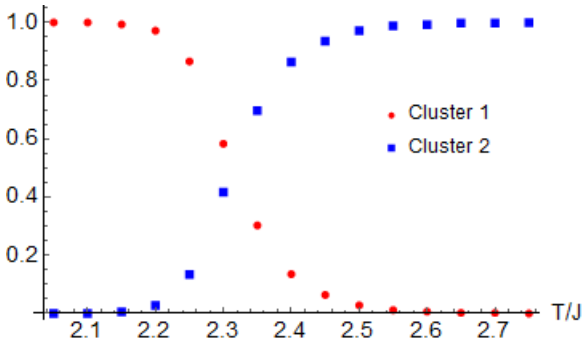
It is important to point out also the main drawbacks of this method. First of all the value of $K$ is required to the user and hence we have to know the number of expected classes a priori, but this is not always possible. Moreover, in systems where only few features are relevant and all the others are noisy variables, clustering could not work as expected and some data could be wrongly classified. In this cases a noise reduction should be performed before clustering, for example a dimensional reduction such as PCA.

### 4.2.1 K-means on the 2D Ising model

In this section we use a $K$-means clustering algorithm to classify spin configurations of the 2D Ising model and to build the phase diagram of the model in the $T - H$ plane (where $H$ is the external magnetic field). The standard approach would be to sample statistical independent equilibrium configurations using the Metropolis-Hastings algorithm at different temperatures and fields, but this is computationally challenging because of the large autocorrelation time, especially at $T \approx T_c$. One should try to use more efficient sampling algorithms, such as the already mentioned Wolff algorithm, that avoids critical slowing down.




(a) Fraction of data assigned to the first (red) and second (blue) clusters at every temperature after performing $K$-means on equilibrium spin configurations.




(b) Fraction of data assigned to the first (red) and second (blue) clusters at every temperature after performing K-means clustering on the absolute value of the first two principal components.

For simplicity we use this approach only at $H = 0$, using the same dataset we have used for testing PCA. The dataset is made by 75000 vectors of $\mathbb{R}^{L^2}$ (5000 samples per temperature at 15 different temperatures), $L = 40$ being the lattice size. We perform a $K$-means clustering with $K = 2$ and compute the fraction of data assigned to the first and second clusters as a function of temperature. Results of this procedure are presented in Fig. 4.13a: low temperature and high temperature samples are correctly separated; samples at $T \approx T_c$ are more difficult to classify.

Better results can be achieved performing $K$-means clustering on the set of principal components. In this case, taking the absolute value of the first two components, we have 75000 vectors of $\mathbb{R}^2$. Looking at Fig. 4.3 it is clear that a direct clustering on principal components (taken with sign) would not provide the correct result. As a consequence, we have to use the $Z_2$ symmetry of the Ising Hamiltonian and take $(|y_1|, |y_2|)$ for every data. The number of samples assigned to each cluster is shown in Fig. 4.13b.

Another interesting approach is suggested in [33], and it consists in clustering of the relaxation curves of magnetization $m(t)$. In this case we don't have to extract an Ising sample once in an autocorrelation time $\tau_m \approx 150L^2$, but once in a Monte Carlo step, which is much smaller ($L^2$ iterations of Metropolis-Hastings algorithm). Moreover, relaxation curves of magnetization show different behaviour when the system relaxes to a paramagnetic or ferromagnetic phase, thus providing a good dataset for constructing the phase diagram. In many physical situations one can not access equilibrium independent states of a system, but one can measure (or at least sample) the dynamical evolution of a physical observable, hence it could be extended to more complicated situations.

Let the time unit be a Monte Carlo step, and let $t_{max} = 500$ be the maximum sampling time; magnetization relaxation curves can be seen as a set of $t_{max}$ numbers that can be stored in a vector of $\mathbb{R}^{t_{max}}$. At every temperature and field the system is prepared in a state with all spins down and $m(0) = -1$, and the Monte Carlo dynamics is performed 20 times for statistics. After performing $K$-means clustering with $K = 2$ (there are two phases), we counted at every $T, H$ the number of samples assigned to the first cluster. Results of this procedure are presented in Fig. 4.14. The phase diagram emerges from the plot. Data deep in ferromagnetic and paramagnetic regions are correctly assigned, while the method is rough in the border between the two phases, especially at high external fields and $T > T_c$.

In Fig. 4.15 we show the components of cluster centroids $\mu_1$ and $\mu_2$ labeled by Monte Carlo time (i.e. the average relaxation curves associated to the two clusters). As expected, data in the second cluster (paramagnetic phase) relax to lower magnetizations than the data in the first one (ferromagnetic).

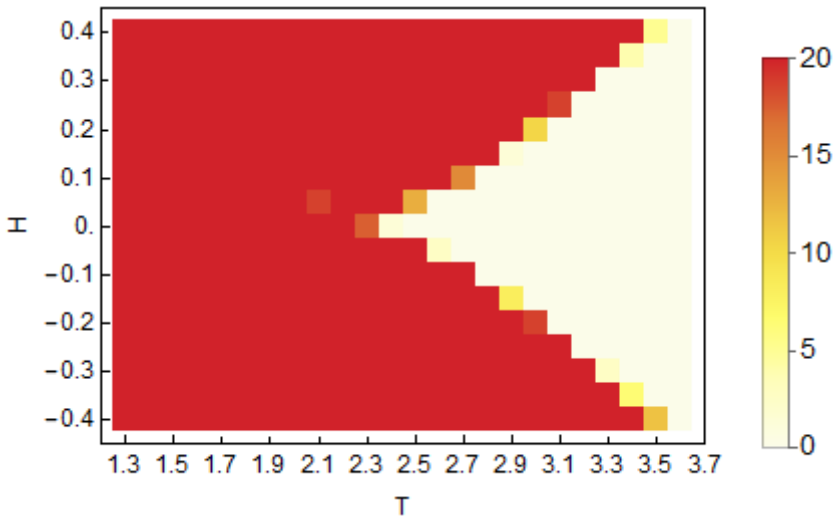



Figure 4.14: Phase diagram of the 2D Ising model ($L = 40$) in the $T - H$ space predicted by $K$-means clustering of magnetization curves. The color scheme represents the number of the 20 curves assigned to the first cluster at every temperature and field.

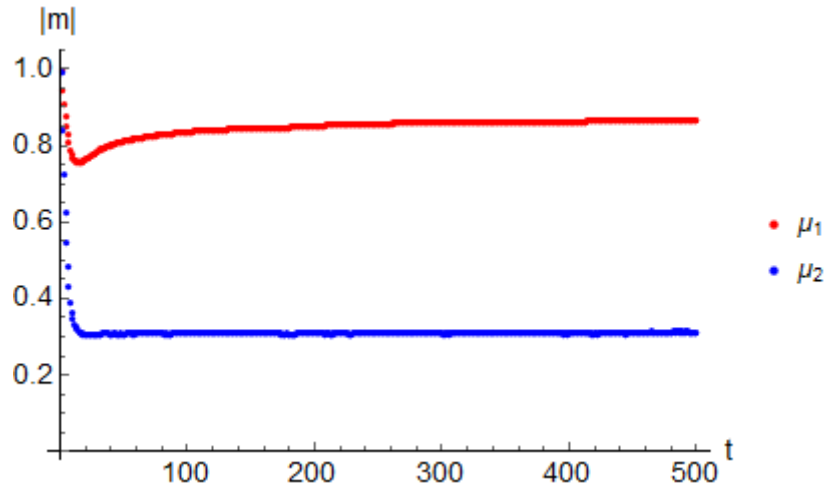



Figure 4.15: Components of the centroids of the two clusters (mean relaxation curves of magnetization).

# Conclusions and outlook

In this work we have proposed physical questions that can be addressed with machine learning, and we have answered these questions using suitable tools. In this section we critically analyze our work, summarizing successes and issues, and proposing ideas for further study.

First of all, we noticed the importance of building a suitable input dataset. In particular, as we investigated critical properties of spin models, we faced the problem of critical slowing down, with consequent large computational effort to sample statistically independent samples. For the simple models that we used here as a benchmark, a good compromise between reasonable sampling time and statistical independence has been found. However, in some contexts a finite size analysis of the system is necessary, and sampling the system in very large lattices is required. This makes it really challenging to sample independent equilibrium configurations, and the input data set is difficult to build. For this reason, a first improvement to this work would be implementing importance sampling Monte Carlo algorithms that prevent critical slowing down, such as the already mentioned Wolff cluster algorithm.

Secondly, we have shown how linear penalized regression is a suitable tool to address the problem of designing Hamiltonians of unknown models. We have analyzed the two most widespread penalizations ($L_1$ and $L_2$), marking analogies and differences, observing in particular how $L_1$ penalization performs a feature selection. We have seen the importance of avoiding overfitting through the validation process, and we have shown how this process can be used to reject unsuitable models. We have used $L_2$ penalization to investigate eight unknown models, and finally found che most suitable coupling constants, and we were able to distinguish between ferromagnetic/antiferromagnetic interactions and long-range/short-range interactions. An interesting step forward on this subject would be investigating non linearizeable Hamiltonians with the same tools, a good starting point being Ising like models coupled to external fields.

After that, we addressed the problem of training classifiers to predict phases of unseen input configurations. We discussed the basic tool of softmax regression, and underlined the problems of abstraction of this approach, noticing the difficulty of recognizing upwards magnetized and downwards magnetized samples both as ordered structures. Motivated by these issues, we developed a feed forward deep neural network, investigating the process of abstraction and improving the classification of the Ising states. We then introduced the more sophisticated structure of convolutional neural networks with the purpose of recognizing topological defects in XY states. The learning process of topological structures, although successfull, is still not clear and it still represents an open research problem that requires further study. Moreover, we have mentioned the recently proposed connection between convolutional networks and the renormalization group theory, a study that we recommend as a further step on this topic.

Finally, we have faced the problem of learning features out of unlabeled data, and we introduced PCA as the simplest tool of unsupervised learning for dimensional reduction. We have explored the efficiency of PCA in extracting order parameters for phase transitions applying it to the Ising, Potts and XY models. Looking at the XY model in particular, we noticed that PCA predicts that spatial correlations between different spins are the relevant feature for the topological phase transition. We have discussed the failure of PCA as a good technique for dimensional reduction when too many principal components have comparable explained variances. In literature, some tools to go beyond PCA have been developed, such as autoencoder networks, t-SNE and Random Boltzmann Machines;

implementing these techniques would be a good starting point for further study.

Besides dimensional reduction, another important unsupervised learning method is clustering. In particular we studied K-means clustering with euclidean distance, and applied it to the Ising model with two approaches (clustering the equilibrium spin configurations or the magnetization relaxation curves). We have seen that clustering the equilibrium spin configurations works in synergy with dimensional reduction, and it is successful in detecting the critical point. Moreover we have seen that dynamical magnetization relaxing curves can be used as a suitable dataset to cluster for obtaining a rough phase diagram of the model when both temperature and magnetic fields are involved in the phase transition.

In conclusion, this work is a review of the very basics of machine learning and provides a collection of physical contexts where these techniques have been successfully applied. Being far from completeness, this work can be regarded as a starting point to familiarize with techniques that could lead to novel results in physics research. In particular, the long term perspective is to apply machine learning in synergy with theoretical efforts to study complex systems of physical interest whose properties are still not completely clear, going beyond the benchmark provided by simple spin models.

# Appendix

In this section we present further documentation about the work. In particular we attach some graphics and codes which are hopefully useful to the interested reader who wants to reproduce our results and eventually extend them to other contexts. For the seek of clearness and simplicity, we do not provide the full codes including data managment, plot styles etc., but only the most relevant routines performing the most interesting tasks. Moreover, the presented codes are explained in detail through captions containing all the details that the user should implement himself-herself to apply the routines.

All the attached codes are written in Wolfram Mathematica 12.0 (some functions could not be available in previous versions), even though some minor tasks were performed with C++ and Python. All the codes are used to perform machine learning tasks, as discussed in the text, while data sampling with Monte Carlo algorithms are not attached. Anyway these are quite standard tasks that are exhaustively exposed in textbooks as Refs. [4, 5].

Detailed information about network training algorithms can be obtained directly either directly running the Wolfram notebook or using the function NetInformation[]. Wolfram also provides online resources where all algorithms are explained in detail.

```
{n, L} = Dimensions[spin];
xprime = Table[spin[[k, i]] * spin[[k, j]], {k, 1, n}, {i, 1, L}, {j, 1, L}];
x = Table[Flatten[xprime[[k]]], {k, 1, n}];
{xtrain, xtest} = Partition[x, n / 2];
{etrain, etest} = Partition[energy, n / 2];

lambda = .01;
wRidge = Flatten[Inverse[(Transpose[xtrain].xtrain + lambda * IdentityMatrix[L * L])].
    Transpose[xtrain].etrain];
MatrixPlot[Partition[wRidge, L]]

net = NetTrain[LinearLayer[1, "Input" → 1600], xtrain → etrain,
  LossFunction → MeanSquaredLossLayer[], Method → {"ADAM", "L2Regularization" → lambda}]
w = Flatten[Normal[NetExtract[net, "Weights"]]];
MatrixPlot[Partition[w, L]]
```

Figure 16: Wolfram code for Ridge regression on the one dimensional Ising model. A number $n$ of onfigurations with $L$ features are stored in a list "spin", while the relative energies are stored in "energy". The first block divides the dataset into training and test sets and computes the spin products $S_i S_j$, so "xtrain" and "xtest" are both lists with dimensions $n/2 \times L^2$. In the second block "wRidge" is the exact solution for Ridge regression with regularizer "lambda" (Eq. 2.5). In the third block we interprete Ridge regression as a simple neural network optimized with "ADAM" algorithm (Ref. [42] for details); here "w" contains the optimized weights. Plots of "w" and "wRidge" are shown as color schemes applied on $L \times L$ matrices.

```
net = NetChain[{
   LinearLayer[3],
   LogisticSigmoid,
   LinearLayer[2],
   SoftmaxLayer["Output" → NetDecoder[{"Class", {1, 0}}]]
  }]
trained = NetTrain[net, data → class]

w1 = NetExtract[trained, {1, "Weights"}] // Normal;
b1 = NetExtract[trained, {1, "Biases"}] // Normal;
y = Table[w1.data[[i]] + b1, {i, 1, n}];
tab1 = Table[{mag[[i]], y[[i, 1]]}, {i, 1, n}];
tab2 = Table[{mag[[i]], y[[i, 2]]}, {i, 1, n}];
tab3 = Table[{mag[[i]], y[[i, 3]]}, {i, 1, n}];
ListPlot[{tab1, tab2, tab3}, PlotRange → All, AxesLabel → {"m", "w1x0+b1"}]
```

Figure 17: Wolfram code for the deep neural network built for classifying states of the Ising model. In the first block the network architecture is built and the network is trained. The list "data" is made by $n$ training samples, with $L^2$ spins each. The $n$-dimensional list "class" classifies the samples' phases: a value in "class" can be either 1 if the corresponding sample is extracted at $T < T_c$ or 0 otherwise. In the second block we extract weights and biases from the hidden layer and plot the output of the layer.

```
Simplify`PWToUnitStep[Piecewise[{
       {# + 2 Pi, -2 Pi ≤ # < -Pi},
       {#, -Pi ≤ # < Pi},
       {# - 2 Pi, Pi ≤ # ≤ 2 Pi}
      }] &[x]] /.
 UnitStep → (1 - Sign[Ramp[-#]] &)
(2 - x) (1 - Sign[Ramp[x]]) + (1 + x) Sign[Ramp[x]];
h[x_] :=
  (1 / 2 Pi) * ((2 π + x) (1 - Sign[Ramp[-2 π - x]]) Sign[Ramp[-π - x]] +
     x (1 - Sign[Ramp[-π - x]]) Sign[Ramp[π - x]] +
     (-2 π + x) (1 - Sign[Ramp[π - x]]) (1 - Sign[Ramp[-2 π + x]]));

trainingset1 = Partition[Table[Partition[data[[i]], L], {i, 1, n}], 1];
validationset1 = Partition[Table[Partition[validation[[i]], L], {i, 1, n1}], 1];
net = NetChain[{
   PaddingLayer[{{0, 0}, {0, 1}, {0, 1}}],
   ConvolutionLayer[4, 2, "Input" → {1, L + 1, L + 1}],
   ElementwiseLayer[h],
   ConvolutionLayer[1, 1],

   PaddingLayer[{{0, 0}, {0, 1}, {0, 1}}],
   ConvolutionLayer[8, 3, "Input" → {1, L + 1, L + 1}],
   ElementwiseLayer["ReLU"],
   ConvolutionLayer[16, 3],
   ElementwiseLayer["ReLU"],
   PoolingLayer[{2, 2}, {2, 2}],
   FlattenLayer[],
   LinearLayer[32],
   ElementwiseLayer["ReLU"],
   LinearLayer[2],
   SoftmaxLayer["Output" → NetDecoder[{"Class", {1, 0}}]]
  }]
trained = NetTrain[net, trainingset1 → class, ValidationSet → validationset1 → classvalidation]

f1 = Normal[NetExtract[trained, {2, "Weights"}]];
f2 = Normal[NetExtract[trained, {4, "Weights"}]];
MatrixForm[f1]
MatrixForm[f2]
```

Figure 18: Wolfram code for the convolutional neural network built for classifying states of the XY model. The first block rewrites the sawtooth function (Eq. 1.36) normalized by $2\pi$ in a suitable form for the function ElementwiseLayer[] and it can be generalized to any function. The function $h(x)$ is defined copying the output of the first block. The list "data" is taken from a file with $n$ rows (number of samples) and $L^2$ columns (number of lattice sites) containing the training dataset. This list is suitably reshaped for compatibility with the network structure and labeled as "trainingset1". The class where any sample belongs is a binary variable (1 if $T < T_{KT}$ and 0 if $T > T_{KT}$) and it is stored in the $n$-dimensional list "class". The network is built up and trained in the third block, while in the last block the learnt filters are extracted and shown in a matrix form. Training is performed with a validation set to reduce overfitting, this is stored in a $n1 \times L^2$ dimensional list "validation" that is suitably reshaped as "validationset1". The architecture is divided in two parts: the first one is the theorized algorithm to transform raw spin configurations into vorticity configurations; the second part is the convolutional structure for binary classification of vorticity dataset suggested in Ref. [32].

```
{n, d} = Dimensions[x];
i = ConstantArray[1, d];
{u, s, v} = SingularValueDecomposition[N[x - Mean[x].i]];
y = x.v;
lambda = s.Transpose[s] / (n - 1);
variance = Table[{i, lambda[[i, i]] / Tr[lambda]}, {i, 1, d}]
ListPlot[variance]
```

Figure 19: Wolfram code for performing a principal component analysis on any dataset imported in list "x". After a singular value decomposition of the input dataset we compute the principal components "y" and plot the behaviour of the explained variance. For large datasets the algorithm is computationally expensive, so we suggest to insert the number of needed principal components as second argument of the function SingularValueDecomposition[].

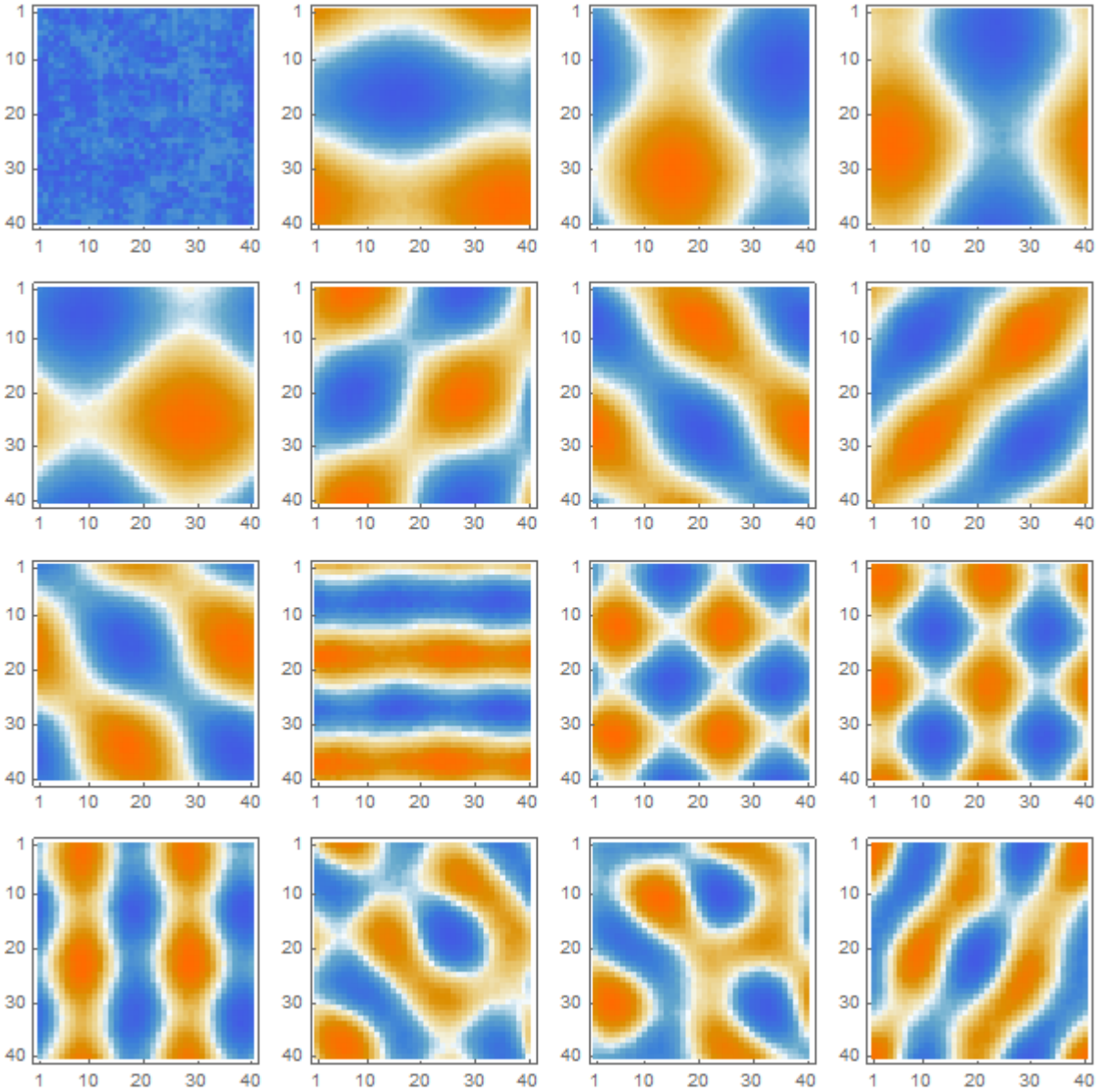


Figure 20: Patterns of weights $\omega_{ij}^{(n)}$ used to compute the first 16 principal components $y_n = \sum_{ij} \omega_{ij}^{(n)} S_{ij}$ of the two dimensional Ising model on a $L = 40$ square lattice. The value of $n$ increases by one unit along the rows. The first two plots have been discussed in Chapter 4, the others are reported here to show their ordered pattern.

# Bibliography


[1] Huang K., *Statistical Mechanics*, Wiley, (1987).

[2] Le Bellac M., *Quantum and statistical field theory*, Clarendon Press - Oxford, (1991).

[3] Baxter R.J., *Exactly solved models in Statistical Physics*, Academic Press - London, (1982).

[4] Robert C.P., Casella G., *Monte Carlo statistical methods*, second edition, Springer - New York (2004).

[5] Giordano N.J., Nakanishi H., *Computational Physics*, second edition, Pearson Prentice Hall - Upped Saddle River, (2006).

[6] Kosterlitz J. M., *Critical properties of XY-model*, J. Phys. C: Solid State Phys., **7**, (1974).

[7] Pankaj Mehta, Ching-Hao Wang, Alexandre G. R. Day, Clint Richardson, Marin Bukov, Charles K. Fisher, David J. Schwab, *A high bias low variance introduction to machine learning for physicists.*

[8] Landau D.P., Wang F., Braz. J. Phys. **34** 2a, (2004).

[9] Katzgraber H. G., *Introduction to Monte Carlo methods*, lecture notes at the third international summer school "Modern Computation Science", Oldenburg University, (2011).

[10] Wu F. Y., *The Potts model*, Rev. Mod. Phys. **54**, 235-268 (1982).

[11] Iino S., Morita S., Sandvik A. W., Kawashima N., J. Phys. Soc. Jpn. **88**, 034006 (2019)

[12] Baxter R. J., Kelland S. B., Wu F. Y., J. Phys. A **9**, 397 (1976).

[13] LeCun Y., Bengio Y., Hinton G., Nature **521**, 436–444, (2015).

[14] Krizhevsky A., Sutskever I., Hinton G., Proc. Advances in Neural Information Processing Systems **25** 1090–1098 (2012).

[15] Celeux G., Forbes F., Peyrard N., *EM-based image segmentation using Potts models with external field*, [Research Report] RR-4456, INRIA, (2002).

[16] Mikolov T., Deoras A., Povey D., Burget L., Cernocky J., Proc. Automatic Speech Recognition and Understanding 196–201 (2011).

[17] Hinton G., Deng L., Dong Y., Dahl G.E., Mohamed A.R., Jaitly N., Senior A., Vanhoucke V., Nguyen P., Sainath T.N., Kingsbury B., IEEE Signal Processing Magazine **29**, 82–97 (2012).

[18] Collobert R., Weston J., Bottou L., Karlen M., Kavukcuoglu K., Kuksa P., J. Mach. Learn. Res. **12**, 2493–2537 (2011).

[19] Sutskever I., Vinyals, O., Le. Q.V., Proc. Advances in Neural Information Processing Systems **27**, 3104–3112 (2014).

[20] Ciodaro T., Deva D., De Seixas J., Damazio D., J. Phys. Conf. Series **368**, 012030 (2012).

[21] Kononenko I., Artificial Intelligence in Medicine, **23**, 1, 89-109, (2001).

[22] Parikh N., Boyd S., Foundations and Trends in Optimization: Vol. 1: No. 3, pp 127-239, (2014).

[23] Beck A., Teboulle M., SIAM journal on imaging sciences, (2009).

[24] Gentle J.E., *Numerical Linear Algebra for Applications in Statistics*. Berlin: Springer-Verlag, (1998).

[25] Nash J.C., *Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation*, 2nd ed. Bristol, England: Adam Hilger, (1990).

[26] Wang L., Phys. Rev. B **94**, 195105 (2016).

[27] Hu W., Singh R.R.P., Scalettar R.T., Phys. Rev. E **95**, 062122 (2017).

[28] Costa N.C., Hu W., Bai Z.J., Scalettar R.T., Singh R.R.P. Phys. Rev. B **96**, 195138 (2017).

[29] Foreman S., Giedt J., Maurice Y., Unmuth-Yockey J., Phys. Rev. E **98**, 052129 (2018).

[30] Wetzel S. J., Phys. Rev. E **96**, 022140 (2017).

[31] Zhang W., Liu J., Wei T.C., Phys. Rev. E **99**, 032142 (2019).

[32] Beach M.J.S., Golubeva A., Melko R.G., Phys. Rev. B **97**, 045207 (2018).

[33] Li L., Yang Y., Zhang D., Ye Z., Jesse S., Kalinin S.V., Vasude R.K., Science Advances Vol. 4, n. 3 **eaap8672** (2018).

[34] Carasquilla J., Melko R.G., Nature Physics **13**, 431–434 (2017).

[35] Jadrich R.B., Lindquist B.A., and Truskett T.M., J. Chem. Phys. **149**, 194109 (2018).

[36] Tanaka A., Tomiya A., J. Phys. Soc. Jpn. **86**, 063001 (2017).

[37] Huembeli P., Dauphin A. and P. Wittek, Phys. Rev. B **97**, 134109 (2018).

[38] Efthymiou S., Beach M.J.S. and Melko R.G., Phys. Rev. B **99**, 075113 (2019).

[39] Mills K., Tamblyn I., Phys. Rev. E **97**, 032119 (2018).

[40] Koch-Janusz M., Ringel Z., Nature Physics **14**, 578–582 (2018).

[41] Li S.H., Wang L., Phys. Rev. Lett. **121**, 260601 (2018).

[42] Kingma D.P., Ba J. - arXiv preprint arXiv:1412.6980, (2014).