Unveiling phase transitions with machine learning

Askery Canabarro,[1,2] Felipe Fernandes Fanchini,[3] André Luiz Malvezzi,[3] Rodrigo Pereira,[1,4] and Rafael Chaves[1,5,*]

[1]*International Institute of Physics, Federal University of Rio Grande do Norte, 59078-970 Natal, Brazil*

[2]*Grupo de Física da Matéria Condensada, Núcleo de Ciências Exatas - NCEx, Campus Arapiraca, Universidade Federal de Alagoas, 57309-005 Arapiraca-AL, Brazil*

[3]*Faculdade de Ciências, Universidade Estadual Paulista, 17033-360 Bauru-SP, Brazil*

[4]*Departamento de Física Teórica e Experimental, Federal University of Rio Grande do Norte, 59078-970 Natal, Brazil*

[5]*School of Science and Technology, Federal University of Rio Grande do Norte, 59078-970 Natal, Brazil*

(Received 12 April 2019; revised manuscript received 1 July 2019; published 22 July 2019)

The classification of phase transitions is a central and challenging task in condensed matter physics. Typically, it relies on the identification of order parameters and the analysis of singularities in the free energy and its derivatives. Here, we propose an alternative framework to identify quantum phase transitions, employing both unsupervised and supervised machine learning techniques. Using the axial next-nearest-neighbor Ising (ANNNI) model as a benchmark, we show how unsupervised learning can detect three phases (ferromagnetic, paramagnetic, and a cluster of the antiphase with the floating phase) as well as two distinct regions within the paramagnetic phase. Employing supervised learning we show that transfer learning becomes possible: a machine trained only with nearest-neighbor interactions can learn to identify a new type of phase occurring when next-nearest-neighbor interactions are introduced. All our results rely on few- and low-dimensional input data (up to twelve lattice sites), thus providing a computational friendly and general framework for the study of phase transitions in many-body systems.

DOI: 10.1103/PhysRevB.100.045129

I. INTRODUCTION

Machine learning (ML) means, essentially, computer programs that improve their performance automatically with increasing exposition to data. The algorithmic improvements over the years combined with faster and more powerful hardware [1–11] allows now the possibility of extracting useful information out of the monumental and ever-expanding amount of data. It is the fastest-growing and most active field in a variety of research areas, ranging from computer science and statistics, to physics, chemistry, biology, medicine and social sciences [12]. In physics, the applications are abundant, including gravitational waves and cosmology [13–23], quantum information [24–26], and in condensed matter physics [27–29] most prominently in the characterization of different phases of matter and their transitions [30–36].

Classifying phase transitions is a central topic in many-body physics and yet a very much open problem, specially in the quantum case due the curse of dimensionality of exponentially growing size of the Hilbert space of quantum systems. In some cases, phase transitions are clearly visible if the relevant local order parameters are known and one looks for nonanalyticities (discontinuities or singularities) in the order parameters or in their derivatives. More generally, however, unconventional transitions such as infinite order (e.g., Kosterlitz-Thouless) transitions, are much harder to be identified. Typically, they appear at considerably large lattice sizes, a demanding computational task which machine learning has been proven to provide a novel approach [30–34]. For instance, neural networks can detect local and global order parameters directly from the raw state configurations [30]. They can also be used to perform transfer learning, for example, to detect transition temperatures of a Hubbard model away from half-filling even though the machine is only trained at half-filling (average density of the lattice sites occupation) [32].

In this paper our aim is to unveil the phase transitions through machine learning, for the first time, in the axial next-nearest-neighbor Ising (ANNNI) model [37–39]. Its relevance stems from the fact that it is the simplest model that combines the effects of quantum fluctuations (induced by the transverse field) and competing, frustrated exchange interactions (the interaction is ferromagnetic for nearest neighbors, but antiferromagnetic for next-nearest neighbors). This combination leads to a rich ground-state phase diagram which has been investigated by various analytical and numerical approaches [40–45]. The ANNNI model finds application, for instance, in explaining the magnetic order in some quasi-one-dimensional spin ladder materials [46]. Moreover, it has recently been used to study dynamical phase transitions [47] as well as the effects of interactions between Majorana edge modes in arrays of Kitaev chains [48,49].

Using the ANNNI model as a benchmark, we propose a machine learning framework employing unsupervised, supervised, and transfer learning approaches. In all cases, the input data to the machine is considerably small and simple, the raw pairwise correlation functions between the spins for lattices up to 12 sites. First, we show how unsupervised learning can detect, with great accuracy, the three main phases of the ANNNI model: ferromagnetic, paramagnetic, and the clustered antiphase/floating phases. As we show, the

*Corresponding author: rchaves@iip.ufrn.br

 ©2019 American Physical Society

unsupervised approach also identifies, at least qualitatively, two regions within the paramagnetic phase associated with commensurate and in-commensurate areas separated by the Peschel-Emery line [50], a subtle change in the correlation functions which is hard to discern by conventional methods using only data from small chains (as we do here). Finally, we show how transfer learning becomes possible: by training the machine with nearest-neighbor interactions only, we can also accurately predict the phase transitions happening at regions including next-nearest-neighbor interactions.

The paper is organized as follows. In Sec. II, we describe in details the ANNNI model. In Sec. III, we provide a succinct but comprehensive overview of the main machine learning concepts and tools we employ in this work (with more technical details presented in the Appendix). In Sec. IV, we present our results: in Sec. IV A, we explain the dataset given as input to the algorithms; in Sec. IV B, we discuss the unsupervised approach followed in Sec. IV C, by an automatized manner to find the best number of clusters; and in Sec. IV D, the supervised/transfer learning part is presented. Finally, in Sec. V, we summarize our findings and discuss their relevance.

II. THE ANNNI MODEL

The axial next-nearest-neighbor Ising (ANNNI) model is defined by the Hamiltonian [37,38]

$$H = -J\sum_{j=1}^{N}\left(\sigma_j^z\sigma_{j+1}^z - \kappa\sigma_j^z\sigma_{j+2}^z + g\sigma_j^x\right). \tag{1}$$

Here, σ_j^a, with $a = x, y, z$, are Pauli matrices that act on the spin-1/2 degree of freedom located at site j of a one-dimensional lattice with N sites and periodic boundary conditions. The coupling constant $J > 0$ of the nearest-neighbor ferromagnetic exchange interaction sets the energy scale (we use $J = 1$), while κ and g are the dimensionless coupling constants associated with the next-nearest-neighbor interaction and the transverse magnetic field, respectively.

The ground-state phase diagram of the ANNNI model exhibits four phases: ferromagnetic, antiphase, paramagnetic, and floating phases. In both the ferromagnetic phase and the antiphase, the $\mathbb{Z}_2$ spin inversion symmetry $\sigma_j^z \mapsto -\sigma_j^z$ of the model is spontaneously broken in the thermodynamic limit $N \to \infty$. However, these two ordered phases have different order parameters. While the ferromagnetic phase is characterized by a uniform spontaneous magnetization, with one of the ground states represented schematically by $\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow$, the antiphase breaks the lattice translational symmetry and has long-range order with a four-site periodicity in the form $\uparrow\uparrow\downarrow\downarrow\uparrow\uparrow\downarrow\downarrow$. On the other hand, the paramagnetic phase is disordered and has a unique ground state that can be pictured as spins pointing predominantly along the direction of the field. Inside the three phases described so far, the energy gap is finite and all correlation functions decay exponentially. By contrast, the floating phase is a critical (gapless) phase with quasi-long-range order, i.e., power-law decay of correlation functions at large distances. This phase is described by a conformal field theory with central charge $c = 1$ [a Luttinger liquid [51] with an emergent U(1) symmetry].

The quantum phase transitions in the ANNNI model are well understood. For $\kappa = 0$, the model is integrable since it reduces to the transverse field Ising model [52]. The latter is exactly solvable by mapping to noninteracting spinless fermions. Along the $\kappa = 0$ line of the phase diagram, a second-order phase transition in the Ising universality class occurs at $g = 1$. It separates the ferromagnetic phase at $g < 1$ from the paramagnetic phase at $g > 1$. Right at the critical point, the energy gap vanishes and the low-energy properties of a long chain are described by a conformal field theory with central charge $c = 1/2$.

Another simplification is obtained by setting $g = 0$. In this case, the model becomes classical in the sense that it only contains σ_j^z operators that commute with one another. For $g = 0$, there is a transition between the ferromagnetic phase at small κ and the antiphase at large κ that occurs exactly at $\kappa = 1/2$. At this classical transition point, the ground-state degeneracy grows exponentially with the system size: any configuration that does not have three consecutive spins pointing in the same direction is a ground state.

For $g \neq 0$ and $\kappa \neq 0$, the model is not integrable and the critical lines have to be determined numerically. In the region $0 \leqslant \kappa \leqslant 1/2$, the Ising transition between paramagnetic and ferromagnetic phases extends from the exactly solvable point $g = 1$, $\kappa = 0$ down to the macroscopically degenerate point $g = 0$, $\kappa = 1/2$. The latter actually becomes a multicritical point at which several critical lines meet. For fixed $\kappa > 1/2$ and increasing $g > 0$, one finds a second-order commensurate-incommensurate (CIC) transition [53] (with dynamical exponent $z = 2$) from the antiphase to the floating phase, followed by a Berezinsky-Kosterlitz-Thouless (BKT) transition from the floating to the paramagnetic phase.

In summary, the ANNNI model has four phases separated by three quantum phase transitions. Approximate expressions for the critical lines in the phase diagram have been obtained by applying perturbation theory in the regime $\kappa < 1/2$ [38] or by fitting numerical results for large chains (obtained by density matrix renormalization group methods [44]) in the regime $\kappa > 1/2$. The critical value of g for the Ising transition for $0 \leqslant \kappa \leqslant 1/2$ is given approximately by [38]

$$g_{\mathrm{I}}(\kappa) \approx \frac{1-\kappa}{\kappa}\left(1 - \sqrt{\frac{1-3\kappa+4\kappa^2}{1-\kappa}}\right). \tag{2}$$

This approximation agrees well with the numerical estimates based on exact diagonalization for small chains [43]. The critical values of g for the CIC and BKT transitions for $1/2 < \kappa \lesssim 3/2$ are approximated respectively by [44]

$$g_{\mathrm{CIC}}(\kappa) \approx 1.05(\kappa - 0.5), \tag{3}$$

$$g_{\mathrm{BKT}}(\kappa) \approx 1.05\sqrt{(\kappa-0.5)(\kappa-0.1)}. \tag{4}$$

In addition, the paramagnetic phase is sometimes divided into two regions, distinguished by the presence of commensurate versus incommensurate oscillations in the exponentially decaying correlations. These two regions are separated by the exactly known Peschel-Emery line [50], which does not correspond to a true phase transition because the energy gap remains finite and there is no symmetry breaking across this

line. The exact expression for the Peschel-Emery line is

$$g_{\mathrm{PE}}(\kappa) = \frac{1}{4\kappa} - \kappa. \tag{5}$$

While the Ising transition is captured correctly if one only has access to numerical results for short chains, cf. Ref. [43], detecting the CIC and BKT transitions using standard approaches requires computing observables for significantly longer chains [44].

III. MACHINE LEARNING REVIEW

Machine learning is defined as algorithms that identify patterns/relations from data without being specifically programed to do so. By using a variety of statistical/analytical methods, learners improve their performance p in solving a task T by just being exposed to experiences E. Heuristically speaking, machine learning occurs whenever $p(T) \propto E$, i.e., the performance in solving task T enhances with increasing training data.

The state-of-the-art of a typical ML project has four somewhat independent components: (i) the dataset $\mathbf{X}$, (ii) a model $m(\mathbf{w})$, (iii) a cost function $J(\mathbf{X}; m(\mathbf{w}))$, and (iv) an optimization procedure. Our aim is to find the best model parameters, $\mathbf{w}$, which minimizes the cost function for the given dataset. This optimization procedure is, generally, numerical and uses variations of the well known gradient descent algorithm. In this manner, by combining distinct ingredients for each component in this recipe, we end up with a myriad of possible machine learning pipelines [1,2].

In machine learning research, there are two main approaches named as supervised and unsupervised learning, related to the experience passed to the learner. The central difference between them is that supervised learning is performed with prior knowledge of the correct output values for a subset of given inputs. In this case, the objective is to find a function that best approximates the relationship between input and output for the dataset. In turn, unsupervised learning, does not have labeled outputs. Its goal is to infer the underlying structure in the dataset, in other words, discover hidden patterns in the data, see Fig. 1 for a pictorial distinction. In this work, in order to propose a general framework, we used both approaches in a complementary manner.

Using the supervised learning as a prototype, one can depict the general lines of a ML project. We first split the dataset into two nonintersecting sets: $\mathbf{X}_{\mathrm{train}}$ and $\mathbf{X}_{\mathrm{test}}$, named training and test sets, respectively. Typically, the test set corresponds to 10%–30% of the dataset. Then, we minimize the cost function with the training set, producing the model, $m(\mathbf{w}^*)$, where $\mathbf{w}^* = \mathrm{argmin}_w\{J(\mathbf{X}_{\mathrm{train}}; m(\mathbf{w}))\}$. We evaluate the cost function of this model in the test set in order to measure its performance with out-of-sample data. In the end, we seek a model that performs adequately in both sets, i.e., a model that generalizes well. In other words, the model works fine with the data we already have as well as with any future data. This quality, called generalization, is at the core of the difference between the machine learning approach and a simple optimization solution using the whole dataset.

The performance of the model is made by evaluating the cost function in the test set. This is generally done by computing the mean square error (MSE) between the prediction made by the model and the known answer (target), $\epsilon_{\mathrm{test}} = \langle J(\mathbf{X}_{\mathrm{test}}; m(\mathbf{w}^*))\rangle$. In a ML project, we are dealing, generally, with complex systems for which we have *a priori* no plausible assumption about the underlying mathematical model. Therefore, it is common to test various types of models $(m_1, m_2, \ldots)$ and compare their performance on the test set to decide which is the most suitable one. In fact, it is even possible to combine them (manually or automatically) in order to achieve better results by reducing bias and variance [1,2,25].

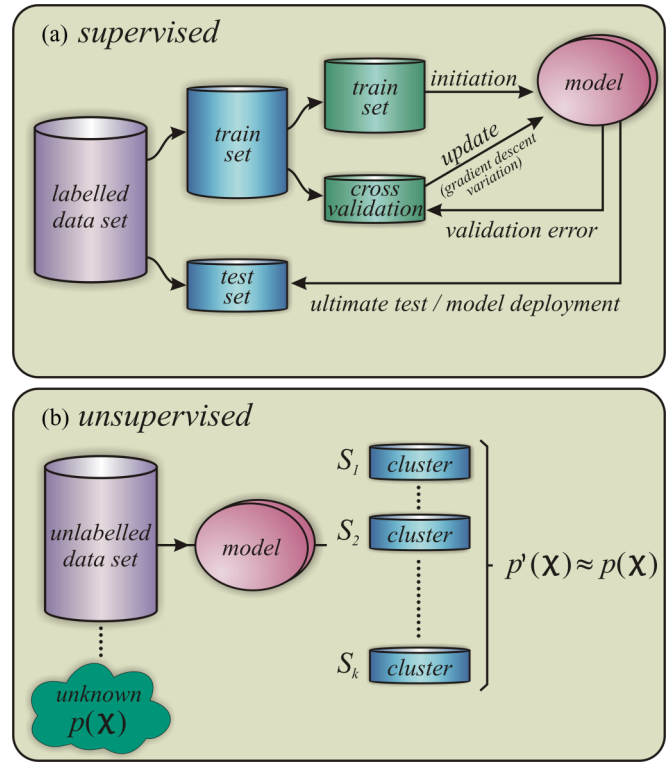

FIG. 1. Schematic representation of machine learning techniques. (a) Supervised learning. (b) Unsupervised learning (clustering).

One should be careful with what is generally called overfitting, that is, some models may present small values for $\epsilon_{\mathrm{train}}$, but $\epsilon_{\mathrm{test}} \gg \epsilon_{\mathrm{train}}$. It happens because some models (often very complex) can deal well with data we already have, however produce large error with unobserved data. Overfitting is a key issue in machine learning and various methods have been developed to reduce the test error, often causing an increase in the training error, but reducing the generalization error as a whole. Making ensemble of multiple models is one of such techniques and as has been already successfully demonstrated [1,25]. On the opposite trend, we might also have underfitting, often happening with very simple models where $\epsilon_{\mathrm{test}} \sim \epsilon_{\mathrm{train}}$ are both large. Although extremely important, the discussion about the bias-variance trade-off is left to the good review in Refs. [1,2].

Overall, the success of a ML project depends on the quality/quantity of available data and also our prior knowledge about the underlying mechanisms of the system. In Appendix, we provide a brief but intuitive description of all

machine learning steps involved in our work. These include the tasks (classification and clustering), the experiences (supervised and unsupervised learning), the machine learning algorithms (multilayer perceptron, random forest, and so on), and also the performance measures. For additional reading and more profound and/or picturesque discussions, we refer to Refs. [1,2] and as well as to Appendix.

IV. MACHINE LEARNING PHASE TRANSITIONS IN THE ANNNI MODEL

A. Our dataset

We have generated the datasets for the correlation functions using exact diagonalization of the Hamiltonian for a chain with periodic boundary conditions. We use the pairwise correlations among all spins in the lattice as the dataset to design our models. Thus the set of observables used is given by $\{\langle\sigma_i^x\sigma_j^x\rangle, \langle\sigma_i^y\sigma_j^y\rangle, \langle\sigma_i^z\sigma_j^z\rangle\}$ with, $j > i$ and $i = [1, N-1]$, where N is the number of spins/qubits in the lattice and $\langle\sigma_i^x\sigma_j^x\rangle = \langle\lambda_0|\sigma_i^x\sigma_j^x|\lambda_0\rangle$ is the expectation value of the spin correlation for the Hamiltonian ground state $|\lambda_0\rangle$ (analogously for $\langle\sigma_i^y\sigma_j^y\rangle$ and $\langle\sigma_i^z\sigma_j^z\rangle$).

We must clarify that we have not computed expectation values of spin operators such as $\langle\sigma_j^z\rangle$. While the latter can be used to define the order parameters of the ferromagnetic phase and the antiphase in the thermodynamic limit, it vanishes identically in the ground state of the Hamiltonian for any finite size system since the ground state respects the $\mathbb{Z}_2$ symmetry of the ANNNI model (the symmetry is spontaneously broken only in the thermodynamic limit). Instead, we analyzed the spin-spin correlation functions, which do include quantum fluctuations and can be used to detect the tendency to a long-range order. For instance, the Fourier transform of the correlation $\langle\sigma_i^z\sigma_j^z\rangle$ defines the static magnetic susceptibility, which exhibits a singularity at zero momentum in the ferromagnetic phase in the thermodynamic limit. Moreover, we would like to add that we have not considered correlation functions which are not diagonal in the spin component, e.g., $\langle\sigma_i^x\sigma_j^z\rangle$, because they vanish due to the $\mathbb{Z}_2$ symmetry. It is easy to see that the number of features is given by $3\sum_{k=1}^{N-1} k$ since that for 8, 10, and 12 sites, we have 84, 135, and 198 features, respectively.

B. Unsupervised approach

Unsupervised learning is the branch of machine learning dealing with data that has not been labeled, classified or categorized. Simply from the features (the components) of the input data, represented by a vector $\mathbf{X}$, one can extract useful properties about it. Quite generally, we seek for the entire probability distribution $p(\mathbf{X})$ that generated and generalizes the dataset. Clustering the data into groups of similar or related examples is a common task in an unsupervised project. Self-labeling is another crucial application of unsupervised learning, opening the possibility of combining unsupervised and supervised algorithms to speed up and/or improve the learning process, an approach known as semisupervised learning (SSL). As we shall demonstrate, using the ANNNI model as a benchmark, unsupervised learning offers a valuable framework to analyze complex phase diagrams even in situations where only few- and low-dimensional input data are available.

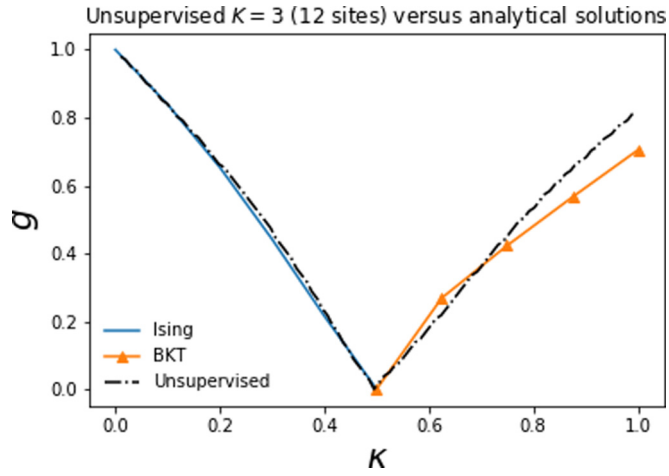

FIG. 2. Comparison among the approximate solutions (2) and (4) and the unsupervised learning trained $N = 12$ sites in the lattice. The Ising transition (2) is almost perfectly reproduced. The machine results for the BKT transition (4) shows a smaller accuracy; nonetheless, it is qualitatively accurate.

Briefly describing, the algorithm is used to partition n samples into K clusters, fixed *a priori*. In this manner, K centroids are defined, one for each cluster. The next step is to associate each point of the dataset to the nearest centroid. Having all points associated to a centroid, we update the centroids to the barycenters of the clusters. So, the K centroids change their position step by step until no more changes are done, i.e., until a convergence is achieved. In other words, centroids do not move more within a predetermined threshold. See Appendix for more details.

For distinct pairs of the coupling parameters $(g;\kappa)$ of the Hamiltonian (1), we explicitly compute all the pairwise spin correlations described in Sec. IV A. Since the computation of correlations is computationally very expensive, the coupling parameters were varied with step size of 10^{-2} in the range $\kappa, g \in [0, 1]$. So, in total, we are training the learner with a modest number of 10 000 examples. Equipped with that, we investigate the capacity of an unsupervised algorithm to retrieve the phase diagram. In Fig. 2, we show the phase diagram produced by using k-means algorithm [54] and focusing on a lattice size $N = 12$. Providing $K = 3$ to the learner (assuming three phases), the algorithm returns the best clustering based on the similarities it could find in the features. Strikingly, given the few data points and the relatively small size of spin chain used to generated the data, it finds three well distinct clusters in very good accordance with the three main phases of the ANNNI model. Indeed, since we imposed to the method the gathering in only three groups, the K-means algorithm detects the ferromagnetic phase, the paramagnetic phase, and a third one which clusters the floating phase with the antiphase. It is quite surprising because the boundary between the paramagnetic and the floating phases is the BKT transition, therefore it detects a transition which is notoriously hard to pinpoint, as the correlation length diverges exponentially at the critical point [55]. Moreover, the unsupervised approach almost perfectly recovers the curve corresponding to the ferromagnetic-paramagnetic transition and its analytical critical value of $g_{\text{crit}} = 1$ (with $\kappa = 0$). As well, it gives very

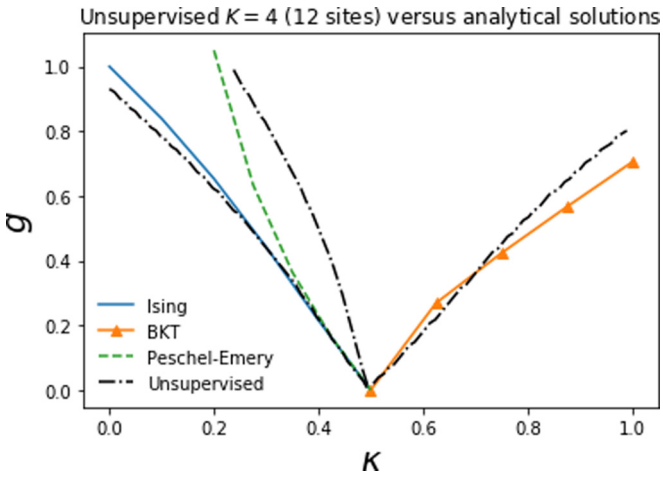

FIG. 3. Comparison among the approximate solutions (2), (4), and (5) with the unsupervised results for $K = 4$. As one can see, at least qualitatively, the machine is finding a new region associated with the Emery-Peschel line (5).

accurate quantitative predictions for the analytical tricritical value $\kappa_{\text{crit}} = 1/2$, at which the transition between the ferromagnetic, paramagnetic and antiphase regions happens.

We have also tested the unsupervised prediction by setting $K = 4$, that is, assuming four phases for the ANNNI model as described in Sec. II. The result is shown in Fig. 3. As we can see, the algorithm does not separate the floating phase and the antiphase but, instead, the new critical line that appears for $K = 4$ divides the paramagnetic phase into two regions which, at least qualitatively, can be identified with the commensurate and incommensurate regions separated by the Peschel-Emery line (5). This result is remarkable because it tells us that machine learning approach manages to detect a subtle change in the correlation functions which is hard to discern by conventional methods using only data for small chains, up to $N = 12$. On the other hand, recall that such change in the correlation function does not correspond to a phase transition in the strict sense.

The results presented in this section indicate that unsupervised learning approach is a good candidate when one knows in advance a good estimate for the number of phases K, as it is requested upfront for various unsupervised algorithms. In the next sections, we show how supervised learning can be used as a validation step for the unsupervised results, also providing surprisingly accurate results. However, we first address the task of how we can use a complementary unsupervised approach to cope with the limitation mentioned above, that is, when one has no *a priori* knowledge of a reasonable number of existing phases.

C. Density-based clustering

To make our framework applicable to cases in which we have no guess of how many phases we can possibly expect, we propose to use a density-based (DB) clustering technique [56,57] to estimate an initial number of clusters. Density clustering makes the intuitive assumption that clusters are defined by regions of space with higher density of data points, meaning that outliers are expected to form regions of low density. The main input in such kind of algorithms is the critical distance ϵ above which a point is taken as an outlier of a given cluster. In fact, it corresponds to the maximum distance between two samples for them to be labeled as in the same neighborhood. One of its most relevant output is the estimated number of distinct labels, that is, the number of clusters/phases. Therefore it can be taken as a complementary technique for the use of the k-means or any other unsupervised approach which requires one to specify the number of clusters expected in the data.

DBSCAN is one of the most common DB clustering algorithms and is known to deal well with data which contains clusters of similar density [57], making it suitable for our case. We use DBSCAN to retrieve the number of clusters we should input in the unsupervised KNN algorithm, thus assuming no prior knowledge of how many phases one expects. For that, we feed the DBSCAN with a critical distance ϵ of the order of the step size used to span the training data, i.e., $\epsilon = 10^{-2}$. As a result, the algorithm returned three clusters as the optimal solution, thus coinciding with the three main phases present in the ANNNI model, precisely those that one can expect to recognize at the small lattice size we have employed. It also coincides with the Elbow curve for estimating the optimal number of clusters, see Appendix for details.

D. Supervised approach

In spite of the clear success of unsupervised ML in identifying the phases in the ANNNI model, a natural question arises. How can we trust the machine predictions in the absence of a more explicit knowledge about the Hamiltonian under scrutiny? Could partial knowledge help in validating the ML results? Typically, limiting cases of a Hamiltonian of interest are simpler and thus more likely to have a known solution. That is precisely the case of the ANNNI model, which for $\kappa = 0$ is fully understood, in particular the fact that at $g = 1$ there is a phase transition between the ferromagnetic and paramagnetic phases. Can a machine trained with such limited knowledge ($\kappa = 0$) make any meaningful predictions to the more general model ($\kappa \geqslant 0$)? The best one can hope for in this situation is that the unsupervised and supervised approaches point out similar solutions, a cross validation enhancing our confidence. In the following, we show that this is indeed possible by investigating supervised learning algorithms as a complementary approach to the unsupervised framework introduced above.

In supervised machine learning, the learner experiences a dataset of features $\mathbf{X}$ and also the target or label vector $\mathbf{y}$, provided by a "teacher," hence the term "supervised." In other words, the learner is presented with example inputs and their known outputs and the aim is to create a general rule that maps inputs to outputs, by generally estimating the conditional probability $p(\mathbf{y}|\mathbf{X})$. One of the main differences between a ML algorithm and a canonical algorithm is that in the second we provide inputs and rules and receive answers and in the first we insert inputs and answers and retrieve rules (see Appendix for more details).

Our aim is to understand whether transfer learning is possible (training with $\kappa = 0$ to predict at regions where $\kappa \geqslant 0$). Both the unsupervised approach as well as the analytical

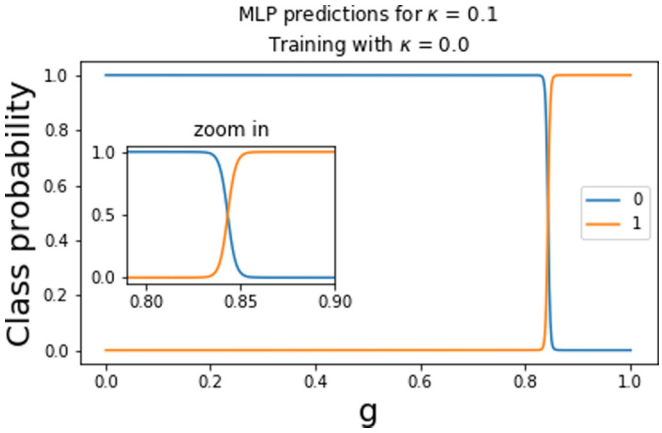

FIG. 4. Detecting the critical transverse magnetic field coupling parameter g at which a phase transition occurs. The machine was trained at $\kappa = 0$ and asked to predict where the transition happens at $\kappa = 0.1$, by considering where the machine is most uncertain, that is, when the probabilities $p_1 = p_2 = 1/2$. Here the ferromagnetic (paramagnetic) phase is labeled as 0 (1).

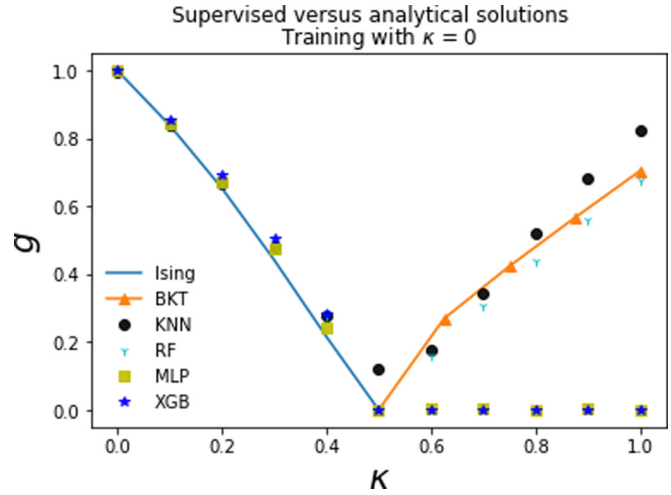

FIG. 5. Phase diagrams produced with diverse ML algorithms when trained only with $\kappa = 0$: KNN (black circles), Random Forest (cyan down stars), Multilayer Perceptron (yellow squares), and Extreme Gradient Boosting (blue diagonal stars) and two different analytical solutions (Ising (solid blue) and BKT (solid orange triangles). All different methods recover the ferro/paramagnetic transition very well while the transition between the paragmanetic and the BKT are only recovered by the KNN and RF methods (see main text for more details).

solution to $\kappa = 0$, point out that a transition occurs at $g \approx 1$. With this information, we train the supervised algorithms with g ranging in the interval [0.5,1.5]. Given that we do not have to vary over κ, we reduce the step size (in comparison with the unsupervised approach) to 10^{-3}, generating an evenly distributed training data with equal number of samples, 500 in each phase (ferromagnetic for $g < 1$ and paramagnetic for $g > 1$). The main drawback of this supervised approach is that it always performs binary classification, known as one-versus-all classification. For instance, for handing writings digits it is similar to the case in which a learner can simply identify whether or not the number 5 has been written.

Motivated by the sound results in its unsupervised version, we first tried the KNN algorithm (vaguely related to the k-mean method) as well as different methods such as the multilayer perceptron (MLP, a deep learning algorithm), random forest (RF) and extreme gradient boosting (XGB). Once the model is trained, we use the same dataset used in Sec. IV B to predict the corresponding phases. Actually, for a given instance X', the trained model, m, returns $m(X') = (p_1, p_2)$, where (p_1, p_2) is a normalized probability vector and the assigned phase corresponds to the component with largest value. To determine when we are facing a transition, we plot both the probability components and check when they cross, as shown in Fig. 4. As can be seen in Fig. 5, the different ML methods successfully recover the left part ($\kappa < 0.5$) of the phase diagram, exactly corresponding to the ferromagnetic/paramagnetic transition over which the machine has be trained for the case $\kappa = 0$. However, what happens as we approach the tricritical point at $\kappa = 0.5$ at which new phases (the antiphase and the floating phase) appears? As can be seen in Fig. 5, near this point the predictions of the different methods start to differ.

To understand what is going on, we highlight that since the machine can only give one out of two answers (the ones it has been trained for), the best it can do is to identify the clustered antiphase/floating phase (here labeled as phase "2") with either the ferromagnetic (phase "0") or the paramagnetic (phase "1") cases. Since the models were never trained for the antiphase, it is a good test to check the learner's ability to classify a new phase as an outlier. For $\kappa \geqslant 0.5$, we are in a region where only phases 0 and 2 are present. So, the best the machine model can do is to output 0 if the phase is indeed 0 and 1 otherwise. As we already remarked, it is a drawback of the supervised approach in comparison to the unsupervised one, but it is still useful to validate the emergence of a new phase, as suggested by the unsupervised technique. In this sense, the KNN and RF methods perform quite well. As seen in Fig. 5 the transition between the paramagnetic and the clustered antiphase/floating is qualitatively recovered even though the machine has never been exposed to these phases before.

In Table I, we present the average ℓ_1 norm for the distinct algorithms taking as benchmark the approximate analytical solutions for the three main phases given by Eqs. (2) and (4).

TABLE I. Performance [average ℓ_1 norm with relation to the analytical approximations given by Eqs. (2) and (3)] computed for the three main phases and different ML approaches. See Appendix for details. Two best ones in boldface.

Technique	average ℓ_1 norm
RF (supervised)	**0.03375(9)**
KNN (supervised)	0.07461(4)
MLP (supervised)	0.18571(4)
XGB (supervised)	0.19507(7)
k-means (12 sites)	**0.03474(4)**
k-means (8 sites)	0.07904(1)
k-means (10 sites)	0.16350(2)

One can observe that the two best approaches are the supervised RF and the unsupervised KNN trained with 12 sites, which reinforces our framework of using the unsupervised and supervised methodologies complementarily. It is worth mentioning that the better performance of the supervised approach is related to the fact that we provided more training data, accounting for a more precise transition, as the step size over g is reduced.

V. DISCUSSION

In this paper, we have proposed a machine learning framework to analyze the phase diagram and phase transitions of quantum Hamiltonians. Taking the ANNNI model as a benchmark, we have shown how the combination of a variety of ML methods can characterize, both qualitatively and quantitatively, the complex phases appearing in this Hamiltonian. First, we demonstrated how the unsupervised approach can recover accurate predictions about the three main phases (ferromagnetic, paramagnetic and the clustered antiphase/floating) of the ANNNI model. It can also recover qualitatively different regions within the paramagnetic phase as described by the Emery-Peschel line, even though there is no true phase transition in the thermodynamic limit. This is remarkable, given that typically this transition needs large lattice sizes to be evidenced. Here, however, we achieve that using comparatively very small lattice sizes (up to 12 spins). Finally, we have also considered supervised/transfer learning and showed how transfer learning becomes possible: by training the machine at $\kappa = 0$ (κ representing the next-neighbor interaction strength) we can also accurately determine the phase transitions happening at $\kappa \geqslant 0$. Surprisingly, the machine trained to distinguish the ferromagnetic and paramagnetic phase is also able to identify the BKT transition, which is notoriously hard to pinpoint, since the correlation length diverges exponentially at the critical point. Moreover, we note that the machine is not simply testing the order parameter because it is able to distinguish between the ferromagnetic phase and the antiphase which are both ordered but with a different order parameter. Indeed, the machine is identifying a new pattern in the data.

Indeed, the standard procedure to infer phase transitions from numerical data for finite systems is to perform a finite size scaling in the order parameters, when the latter are known. If one computes an order parameter for any finite system, the result is a smooth function of the control parameter, with the singularity at the critical point becoming gradually more pronounced as one increases the system size. However, one of the questions we set out to address in this work was precisely whether a machine learning algorithm is able to infer the existence of different phases from limited numerical data for a relatively small system, without comparing different systems sizes. This is possible because hints of the phases that exist in the thermodynamic limit are already present in a finite size system, only the boundaries between the phases will change as the system size grows. Clearly, the algorithm does not employ the same methods that physicists have grown used to in the study of phase transitions. Nevertheless, it manages to extract enough information from the correlation functions to distinguish most of the phases. We should mention, however, that this success is not absolute, as we find that the algorithm indicates a "transition" near the Peschel-Emery line, which is known to be a crossover rather than an actual phase transition in the ANNNI model.

Taken together, these results suggest that the ML investigation should start with the unsupervised techniques (DBSCAN + Clustering), as it is a great initial exploratory entry point in situations where it is either impossible or impractical for a human or an ordinary algorithm to propose trends and/or have insights with the raw data. After some hypotheses are collected with the unsupervised approach, one should now proceed to get more data and apply supervised techniques as to consolidate the unsupervised outputs. See Appendix for more discussions.

Overall, we see that both the unsupervised and the various supervised machine learning predictions are in very good agreement. Not only do they recover similar critical lines but also discover the multicritical point of the phase diagram. Clearly, we can only say that because the precise results are known for the ANNNI model. The general problem of mapping out phase diagrams of quantum many-body systems is still very much open, even more so with the discovery of topological phases. The methods investigated here may contribute to advancing this field and hope to motivate the application of this framework to Hamiltonians where the different phases have not yet been completely sorted out. If as it happens for the ANNNI model, if all this multitude of evidence obtained by considerably different ML methods point out similar predictions, this arguably give us good confidence about the correctness of the results. The predictions given by the ML approach we propose here can be seen as guide, an initial educated guess of regions in the space of physical parameters where phase transitions could be happening.

ACKNOWLEDGMENTS

The authors acknowledge the Brazilian ministries MEC and MCTIC, funding agency CNPq (AC's Universal Grant No. 423713/2016–7, FFF's Universal Grant No. 409309/2018–4, RC's Grants No. 307172/2017–1, and No. 406574/2018–9 and INCT-IQ), FAPESP (FFF's Grant No. 2019/05445–7) and UFAL (AC's paid license for scientific cooperation at UFRN). This work was supported by the Serrapilheira Institute (Grant No. Serra-1708-15763) and the John Templeton Foundation via the Grant Q-CAUSAl No 61084.

APPENDIX A: THE EXPERIENCE E: SUPERVISED AND UNSUPERVISED LEARNING

Within machine learning research, there are two major methods termed as supervised and unsupervised learning. In a nutshell, the key difference between them is that supervised learning is performed using a firsthand information, i.e., one has ("the teacher," hence the term "supervised") prior knowledge of the output values for all and every input samples. Here, the objective is to find/learn a function that best ciphers the relationship between input and output. In contrast, unsupervised learning deals with unlabeled outputs and its goal is to infer the natural structure present within the dataset.

In a supervised learning project, the learner experiences data points of features $\mathbf{X}$ and also the corresponding target or label vector $\mathbf{y}$, aiming to create a universal rule that maps inputs to outputs, by generally estimating the conditional probability $p(\mathbf{y}|\mathbf{X})$. Typical supervised learning tasks are: (i) classification, when one seeks to map input to discrete output (labels) or (ii) regression, when one wants to map input to a continuous output set. Common supervised learning algorithms include: Logistic Regression, Naive Bayes, Support Vector Machines, Decision Tree, Extreme Gradient Boosting (XGB), Random Forests, and Artificial Neural Networks. Later, we provide more details of the ones used in this work.

In unsupervised learning, as there is no teacher to label, classify or categorize the input data, the learner must grasp by himself how to deal with the data. Simply from the features (the components) of the input data, represented by a vector $\mathbf{X}$, one can extract useful properties about it. Quite generally, we seek for the entire probability distribution $p(\mathbf{X})$ that generated and generalizes the dataset, as we have pointed out in the main text. The most usual tasks of an unsupervised learning project are: (i) clustering, (ii) representation learning [58] and dimension reduction, and (iii) density estimation. Although we focus on clustering in this work, in all of these tasks, we aim to learn the innate structure of our unlabelled data. Some popular algorithms are: K-means for clustering; Principal Component Analysis (PCA), and autoencoders for representation learning and dimension reduction; and Kernel Density Estimation for density estimation. Since no labels are given, there is no specific way to compare model performance in most unsupervised learning methods. In fact, it is even hard to define technically Unsupervised Learning, check [1,2] for deeper discussions.

As briefly mentioned in the main text, unsupervised learning is very useful in exploratory analysis and integrated many ML pipelines once it can automatically identify structure in data. For example, given the task of labeling or inserting caption for the tantamount of images available on the world web, unsupervised methods would be (and in fact is) a great starting point for this task, as those used by Google's Conceptual Captions Team. In situations like this, it is impossible or at least impractical for a human or even a regular algorithm to perform such task.

Machine learning is a method used to build complex models to make predictions in problems difficult to solve with conventional programs. It may as well shed new light on how intelligence works. However, it is worth mentioning that this is not, in general, the main purpose behind machine learning techniques given that by "learning" it is usually meant the skill to perform the task better and better, not necessarily how it is learning the task itself. For instance, in a binary classification task, e.g., "cat versus dog," the program does not learn what a dog is in the same sense as human does, but it learns how to differentiate it from a cat. In this manner, it only learns the following truisms: (i) a dog is a dog and (ii) a dog is not a cat (and similar statements for what a cat is). Also, it is important to emphasize that a highly specialized learning process does not imply, necessarily, a better instruction. For example, a more flexible process may, in fact, be more efficient to achieve an efficient transfer learning.

APPENDIX B: TASKS: CLASSIFICATION AND CLUSTERING

A machine learning task is specified by how the learner processes a given set of input data $\mathbf{X} \in \mathbb{R}^n$. Some usual ML tasks are: classification, regression, clustering, transcription, machine translation, anomaly detection, and so on. Here, we quickly describe the two kinds of tasks we use in this work: classification and clustering.

1. Classification

Here, the program must designate in which class an input instance should be classified, given K possible categories. The algorithm retrieves a function $f : \mathbb{R}^n \to \{1, \ldots, k\}$, where K is a finite and commonly preset integer number. So, for a given input vector $\mathbf{X}$, the model returns a target vector $\mathbf{y} = f(\mathbf{X})$. Typically, f returns a normalized probability distribution over the K classes and the proposed class is that with highest probability. Referring to the main text, this is the case for all the supervised algorithms.

2. Clustering

Clustering is one of the most important unsupervised learning problem. As the other problems of this approach, it aims to identify a structure in the unlabelled dataset. A loose definition of clustering could be "the process of organizing objects into groups whose members are similar in some way." A cluster is therefore a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters. Later in this Appendix, we provide more detail about k-means as well as density-based clustering. However, for a more technical definition, please check Ref. [56]. For a formal mathematical description, we refer the reader to Ref. [59].

APPENDIX C: THE PERFORMANCE p

One important aspects of supervised learning which makes it different from an optimization algorithm is *generalization*, as mentioned in Sec. III. It means that we project the algorithm to perform well both on seen (train set) and unseen (test set) inputs. This is accomplished by measuring the performance on both the sets. The test set normally corresponds to 10%–30% of the available data.

For a binary classification tasks (as the ones we dealt with in this work), one practical measure is the accuracy score which is the proportion of correct predictions produced by the algorithm. For the classification task in this work, we were satisfied with a specific model, i.e., the model was taken to be optimal, when both the train and test sets accuracy score were higher than 99.5%. It is worth mentioning that we used these well-trained models for $\kappa = 0$ when we seek to achieve an efficient transfer learning. In other words, when we tried to used these models for $\kappa > 0$.

As already addressed in the main text, the optimal model is found by minimizing a cost function. However, the ideal cost function varies from project to project and from algorithm to algorithm. For all supervised techniques, we used the default cost function of the PYTHON scikit-learn package for

the implementation of a given algorithm [60]. Therefore we present the cost functions case by case in the next subsection. However, to present the performance in Table I, we use the L_1 error due to its straightforward interpretation for model comparison.

APPENDIX D: ALGORITHMS

All algorithms used in this work are native to the PYTHON Scikit-Learn package, see Ref. [60] for more details. Here, we provide a description of the used methods, highlighting the parameters and hyper-parameters used to calibrate the models and consequently generate the figures.

1. *k*-means clustering

The k-means is an unsupervised learning algorithm. It searches for k clusters within an unlabeled multidimensional dataset, where the centers of the clusters are taken to be the arithmetic mean of all the points belonging to the respective cluster, hence the term k-means. Clustering stands for finding groups of similar objects while keeping dissimilar objects in different groups [56]. When convergence is reached, each point is closer to its own cluster center than to other cluster centers. It can be seen as a vector quantization (discrete output), once its final product it to assert labels to the entire dataset.

The dataset is, generally, a set of d-dimensional real-valued points sampled from some unknown probability distribution and the dissimilarity function is often the Euclidean distance. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster and providing a meaningful grouping of the dataset. This results in a partitioning of the data space into Voronoi cells, as schematically represented in Fig. 6.

Given a set of n observations $(X_1, X_2, \ldots, X_n)$, where $X_i \in \mathbb{R}^d$, k-means clustering seeks to partition the n observations into $k(k \leqslant n)$ sets $S = S_1, S_2, \ldots, S_k$ so as to minimize the within-cluster sum of squares, which happens to be the variance. Formally, the objective is to find

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}\in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \,\mathrm{Var}\, S_i, \quad \text{(D1)}$$

where $\boldsymbol{\mu}_j$ is the mean of the points in S_i. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{\mathbf{x},\mathbf{y}\in S_i} \|\mathbf{x} - \mathbf{y}\|^2. \quad \text{(D2)}$$

The algorithm runs as follows: (1) (randomly) initialize the k clusters centroids $\boldsymbol{\mu}_j$; (2) [repeat until convergence] (a) for every observation X_i assign it to the nearest cluster and (b) update the cluster centroid. Check Ref. [59] for detailed mathematical demonstration of asymptotic behavior and convergence. Intuitively, one seeks to minimize the within-group dissimilarity and maximize the between-group dissimilarity for a predefined number of clusters k and dissimilarity function. From a statistical perspective, this approach can be used to retrieve k probability densities p_i. Assuming that they belong to the same parametric family (Gaussian, Cauchy, and so on), one can take the unknown distribution p as a mixture of the k distribution p_i, as depicted in Fig. 2(b).

We have used the library *sklearn.cluster.KMeans* which uses some new initialization technique for speeding up the convergence time. The main input parameter is the number of clusters. Overall, k-means is a very fast algorithm. Check the library for the details and the default parameters [60].

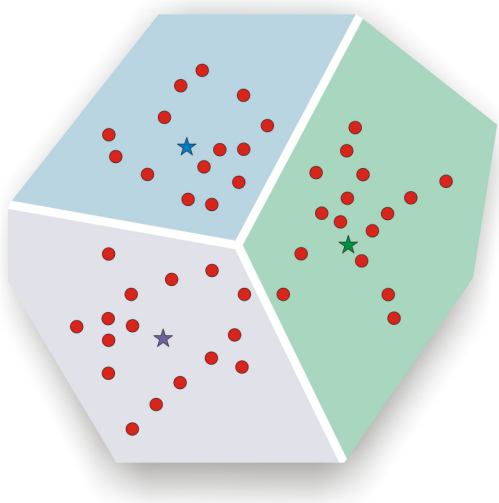

FIG. 6. Schematic representation of the k-means clustering.

2. Density-based (DB) clustering

In density-based clustering methods the clusters are taken to be high-density regions of the probability distribution, giving reason for its name [56]. Here, the number of clusters is not required as an input parameter. Also, the unknown probability function is not considered a mixture of the probability functions of the clusters. It is said to be a nonparametric approach. Heuristically, DB clustering corresponds to partitioning group of high density points (core points) separated by contiguous region of low density points (outliers).

In this work, we used the Density-Based Spatial Clustering (DBSCAN) using the scikit-learn package *sklearn.cluster.DBSCAN*. The algorithm is fast and very good for dataset which contains clusters of similar density [60]. The most important input parameter is the maximum distance between two points for them to be considered as in the same neighborhood, ϵ. We used $\epsilon = 10^{-2}$, as discussed in the main text, and the main output is the number of clusters. In our case, it reported three clusters, which is a sound estimation and fits well the Elbow curve for predicting the optimal k for the k-means clustering. The Elbow method corresponds to a analysis designed to help find the appropriate number of clusters in a dataset. It plots the sum of the negative (Score method in scikit-learn KMeans package) of the clusters variance as a function of the assumed number of clusters. One should pick a number of clusters so that adding another cluster does not give much better modeling of the dataset. In Fig. 7, it is $k = 3$.

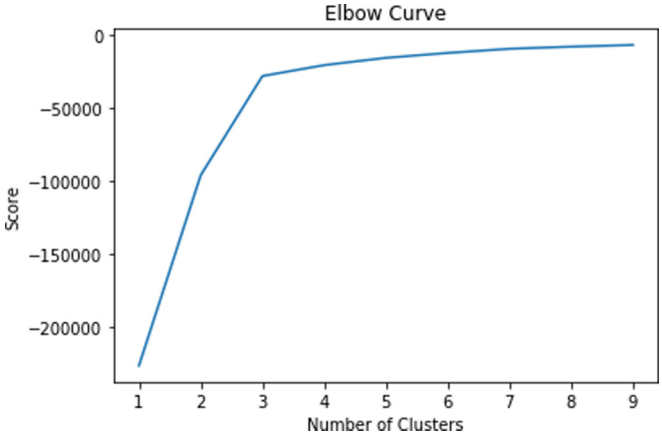

FIG. 7. Elbow curve for the estimation of the best k for k-means clustering method.

3. k-nearest neighbors

The k-means algorithm has a vague relationship to the K-nearest neighbor (KNN) classifier [4] (not for the k in the name, but because k-means classifier is very similar to KNN with $K = 1$), so it was our first weapon when switching from unsupervised to supervised learning, given the success of the previous one. Roughly speaking, the KNN method considers that similar objects are close to each other.

KNN is one of the simplest algorithms in ML using basically the notion of distance in its construction. It is a neighbor-based classification where we perform instance-based learning or nongeneralizing learning, meaning that it does not construct a model, but simply stores instances of the training data. Classification is made from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.

The algorithm runs as follows: (1) set K value; (2) for every instance X (query example) in the dataset: (i) calculate the distance between the query example and true output, (ii) add the distance and the corresponding index of the query example to an ordered collection; (3) sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances; (4) pick the first K entries from the sorted collection; (5) get the labels of the selected K entries; and (6) return the mode of the K labels (classification tasks).

The code was made using the package *sklearn.neighbors. KNeighborsClassifier* [60]. The important parameter used was *n_neighbors*=7 ($K = 7$). The default is the Euclidean distance, although it is possible to choose others such as Minkowsky or Manhattan distance.

4. Random Forest

Random Forest [3] is a powerful ensemble method widely used for hard and complex classification tasks. It consists of a multitude of Decision Trees. It outputs the most voted (mode) class of the individual decision trees. It is intuitive that training various models and taking their weighted predictions could be beneficial, once it mimics the ancient idea behind the "wisdom of the crowds." However, let us first define the Decision Tree learning.

Decision Tree learning consists in the construction of a decision tree from labeled training instances. The root node is the training data itself, the branches nodes are the output (Boolean output) of tests for the attributes and leaves nodes are the estimated class. Therefore, for a given input X_1, a label y_1 is predicted after percolating from the root up to the leaf. In general, isolated, individual Decision Tree algorithm is a weak learner, producing high-bias error.

To implement a combination of Decision Trees, we first partition the training set in M smaller subsets $B_1, B_2, \ldots, B_M$. The subsets are randomly chosen (and may be repeated). If the subsets are large enough for training a specific learner, they can be aggregated to create an ensemble predictor. For classification tasks, we take a majority vote of all the predictions. This process was introduced in Ref. [5] and is called *BAGGing*, acronym for Bootstrap AGGregation. This can be demonstrated to reduce variance (out-of-sample error) without increasing the bias (in-sample error) [2]. In bagging, the contribution of the predictor is taken to be equal.

The key idea of Random Forest is to perform subsets of the features. Now, the trees randomly choose a k number, of N total features with $k < N$. This bagging of features reduces the correlation between the various Decision Trees, contributing to better modeling. It can be formally demonstrated that a large amount of uncorrelated *a priori* weak predictor can be aggregated to reduce variance [2].

The code was made using the package *sklearn.ensemble. RandomForestClassifier* [60]. The important parameters used were *max_depth*=5, *n_estimators*=100.

5. Extreme Gradient Boosting (XGB)

Extreme Gradient Boosting (XGBoost) [61] is another powerful ensemble technique. It is build upon the idea of the Boosting method. Unlike bagging, in boosting k classifier contributes with different weights α_k. So, we have here a weighted sum of the classifier predictors. It makes sense, since one can imagine that, for instance, the "opinion" of a better classifier (measured on the test set) should contribute more in comparison to weaker classifier, which is sometimes termed as "autocratic" approach in contrast with the "democratic" view in the bagging method.

In XGB, the idea is even more sophisticated. Here, the addition of new Decision Trees is done if it contributes to minimizing a cost function of the ensemble. In this manner, XGB provides a clever way to map the gradient to Decision Trees. The technical details are far beyond the scope of this Appendix, but the implication by means of the Scikit-Learn API *XGBClassifier* is trivial. The main parameters used were *max_depth*=5, *n_estimators*=100.

6. Multilayer Perceptron (MLP)

Multilayer Perceptrons are the pillar of what is termed as deep learning modeling. It is a sophisticated supervised learning technique in the class of artificial feed-forward neural networks (or simply neural nets) used to approximate an unknown function $f(\mathbf{X})$ by $f^*(\mathbf{X};\boldsymbol{\theta})$, which maps an input $\mathbf{X}$ to an output $\mathbf{y}$. Once the learning process is concluded back, it retrieves the optimal $\boldsymbol{\theta}$ [2,7]. In general, the larger the dimension of the vector $\boldsymbol{\theta}$ (the number of

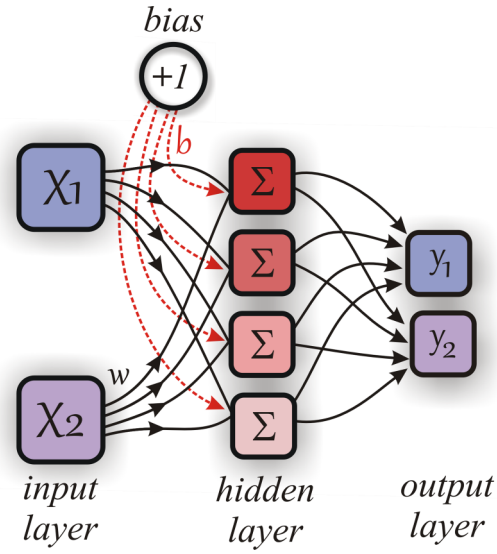

FIG. 8. Architecture of an artificial neural network composed of three layers with one hidden layer. The nodes stand for neurons outputs and the solid (dashed) arrows represent neuron-specific weight (bias). Each neuron processes the incoming signals by using the same activation function Σ.

parameters), the better the approximation f^* for complex tasks can be.

They were initially formulated to emulate a nervous system (the reason for the term "neural") composed of a multitude of basic units or neurons stacked into layers, in which the output of one layer is the input for the succeeding layer. They are said to be feed-forward in virtue of the one-way flow from the input layer to the output layer, see Fig. 8.

The connections between layers are performed by means of the combination of the inputs provided by all neurons in the previous layers (together with a re-centering bias) and all neurons in current layer i (see solid and dashed arrows in Fig. 8) given by

$$\mathbf{z}_i = \mathbf{w}_i \cdot \mathbf{x} + \mathbf{b}_i, \tag{D3}$$

where $\mathbf{x}$ is the input vector (output of previous layer), $\mathbf{w}_i$ and b_i are the vectors of neuron-specific weights and bias for the i layer, respectively [2]. The bias acts similarly to the non-null intercept in a linear regression problem, augmenting the space of tentative solutions, enhancing the power of representation (or expressivity) of the model [2].

Paradoxically, for first layer $\mathbf{z}_i = \mathbf{X}$, i.e., the input layer just outputs the features themselves to the next layer. For the hidden layer, each neuron j outputs a scalar after processing a nonlinear transformation of inputs it previously received. For instance, if the activation function is a (Heaviside) step function, a given neuron receives various signals from the other neurons in the previous layer and "decides" if it should activate (add up of the signals is greater than zero, outputting 1) or not (add up of the signals is less than zero, outputting 0). This step function was the original nonlinearity implemented in the MLP, hence the denomination "activation function." Currently, usual activation functions are, for instance, the logistic function $[\Sigma(x) = 1/(1 + \exp^{-x})]$ and the hyperbolic tangent $[\Sigma(x) = \tanh x]$. The number of hidden layers and activate function are hyperarameters that one can adjust to achieve better predictions. In this work, we used the rectified linear unit (ReLU) function, $\Sigma(x) = \max(0, x)$. Not only because of increasing accuracy, but also due to the reduction in the computational time, once it has be shown to be optimal in computing the gradient which updates the parameter of the model $\mathbf{w}$ and $\mathbf{b}$ via the back-propagation method [11]. This is easily done and requires one to set the "activation = relu" in the package *sklearn.neural_network.MLPClassifier* [60]. Moreover, we used two hidden layers of 100 neurons each in our research. The middle layers are "hidden" layers because it is not known what they should output for the next layer in order to reach the aim of finding a value $y*$ close to y for each X in $\mathbf{X}$. When the output layer is reached, it performs, in general, a simple logistic regression or softmax for classification tasks or linear regression for regression problems.

Given a cost function $J(\boldsymbol{\theta})$, for instance,

$$J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{\mathbf{x} \in \mathbf{X}^{\text{train}}} (\mathbf{y}^{\text{train}} - f^*(\mathbf{x}; \boldsymbol{\theta}))^2, \tag{D4}$$

where m represents the training set size, we encounter the hard problem of computing the gradient, $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$, of a high dimensional and complex function. Even for a model with a few hidden layers of hundreds of neurons we have thousands or millions of parameters. The best $\boldsymbol{\theta}$ returns the values of the weights $\mathbf{w}$ (solid arrows in Fig. 8) and the bias $\mathbf{b}$ (dashed arrows in in Fig. 8) which give the best generalization. One overcomes this computational cost by minimizing the error in the direction from the output layer to the input layer with the back-propagation algorithm and its variants. In our codes, we used the *Adam* algorithm by just setting "solver = adam" in the scikit-learn MLP package, see Ref. [9] for details.

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, 2016).

[2] P. Mehta, M. Bukov, C.-H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, A high-bias, low-variance introduction to machine learning for physicists, Phys. Rep. **810**, 1 (2019).

[3] L. Breiman, Random forests, Mach. Learn. **45**, 5 (2001).

[4] N. S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, Am. Stat. **46**, 175 (1992).

[5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees* (Wadsworth International Group, Belmont, CA, 1984).

[6] P. Geurts, D. Ernst, and L. Wehenkel, Extremely randomized trees, Mach. Learn. **63**, 3 (2006).

[7] F. Rosenblatt, The perceptron: A probabilistic model for information storage and organization in the brain, Psychol. Rev. **65**, 386 (1958).

[8] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore, Evaluation of a tree-based pipeline optimization tool for automating data science, in *Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO '16* (ACM, New York, NY, 2016), pp. 485–492.

[9] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[10] X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga, Ensemble deep learning for regression and time series forecasting, in *2014 IEEE Symposium on Computational Intelligence in Ensemble Learning (CIEL)* (IEEE, New York, 2014), pp. 1–6.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, Nature (London) **323**, 533 (1986).

[12] M. I. Jordan and T. M. Mitchell, Machine learning: Trends, perspectives, and prospects, Science **349**, 255 (2015).

[13] H. M. Kamdar, M. J. Turk, and R. J. Brunner, Machine learning and cosmological simulations–i. semi-analytical models, Mon. Not. R. Astron. Soc. **455**, 642 (2015).

[14] H. M. Kamdar, M. J. Turk, and R. J. Brunner, Machine learning and cosmological simulations–ii. hydrodynamical simulations, Mon. Not. R. Astron. Soc. **457**, 1162 (2016).

[15] J. D. Kelleher, B. Mac Namee, and A. D'Arcy, *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies* (MIT Press, Cambridge, 2015).

[16] M. Lochner, J. D. McEwen, H. V. Peiris, O. Lahav, and M. K. Winter, Photometric supernova classification with machine learning, Astrophys. J. Suppl. Ser. **225**, 31 (2016).

[17] T. Charnock and A. Moss, Supernovae: Photometric classification of supernovae, Astrophysics Source Code Library (2017).

[18] R. Biswas, L. Blackburn, J. Cao, R. Essick, K. A. Hodge, E. Katsavounidis, K. Kim, Y.-M. Kim, E.-O. Le Bigot, C.-H. Lee *et al.*, Application of machine learning algorithms to the study of noise artifacts in gravitational-wave data, Phys. Rev. D **88**, 062003 (2013).

[19] M. Carrillo, J. A. González, M. Gracia-Linares, and F. S. Guzmán, Time series analysis of gravitational wave signals using neural networks, J. Phys.: Conf. Ser. **654**, 012001 (2015).

[20] A. Gauci, K. Z. Adami, and J. Abela, Machine learning for galaxy morphology classification, arXiv:1005.0390.

[21] N. M. Ball, J. Loveday, M. Fukugita, O. Nakamura, S. Okamura, J. Brinkmann, and R. J. Brunner, Galaxy types in the sloan digital sky survey using supervised artificial neural networks, Mon. Not. R. Astron. Soc. **348**, 1038 (2004).

[22] M. Banerji, O. Lahav, C. J. Lintott, F. B. Abdalla, K. Schawinski, S. P. Bamford, D. Andreescu, P. Murray, M. J. Raddick, A. Slosar *et al.*, Galaxy zoo: Reproducing galaxy morphologies via machine learning, Mon. Not. R. Astron. Soc. **406**, 342 (2010).

[23] C. E. Petrillo, C. Tortora, S. Chatterjee, G. Vernardos, L. V. E. Koopmans, G. Verdoes Kleijn, N. R. Napolitano, G. Covone, P. Schneider, A. Grado, and J. McFarland, Finding strong gravitational lenses in the kilo degree survey with convolutional neural networks, Mon. Not. Royal Astron. Soc. **472**, 1129 (2017).

[24] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo, Neural-network quantum state tomography, Nat. Phys. **14**, 447 (2018).

[25] A. Canabarro, S. Brito, and R. Chaves, Machine Learning Non-Local Correlations, Phys. Rev. Lett. **122**, 200401 (2019).

[26] R. Iten, T. Metger, H. Wilming, L. Del Rio, and R. Renner, Discovering physical concepts with neural networks, arXiv:1807.10300.

[27] G. Carleo and M. Troyer, Solving the quantum many-body problem with artificial neural networks, Science **355**, 602 (2017).

[28] G. Torlai and R. G. Melko, Learning thermodynamics with boltzmann machines, Phys. Rev. B **94**, 165134 (2016).

[29] L. M. Ghiringhelli, J. Vybiral, S. V. Levchenko, C. Draxl, and M. Scheffler, Big Data of Materials Science: Critical Role of the Descriptor, Phys. Rev. Lett. **114**, 105503 (2015).

[30] J. Carrasquilla and R. G. Melko, Machine learning phases of matter, Nat. Phys. **13**, 431 (2017).

[31] P. Broecker, J. Carrasquilla, R. G. Melko, and S. Trebst, Machine learning quantum phases of matter beyond the fermion sign problem, Sci. Rep. **7**, 8823 (2017).

[32] K. Ch'ng, J. Carrasquilla, R. G. Melko, and E. Khatami, Machine Learning Phases of Strongly Correlated Fermions, Phys. Rev. X **7**, 031038 (2017).

[33] D.-L. Deng, X. Li, and S. D. Sarma, Machine learning topological states, Phys. Rev. B **96**, 195145 (2017).

[34] P. Huembeli, A. Dauphin, and P. Wittek, Identifying quantum phase transitions with adversarial neural networks, Phys. Rev. B **97**, 134109 (2018).

[35] Y. Zhang and E.-A. Kim, Quantum Loop Topography for Machine Learning, Phys. Rev. Lett. **118**, 216401 (2017).

[36] X.-Y. Dong, F. Pollmann, and X.-F. Zhang, Machine learning of quantum phase transitions, Phys. Rev. B **99**, 121104(R) (2019).

[37] W. Selke, The ANNNI model — theoretical analysis and experimental application, Phys. Rep. **170**, 213 (1988).

[38] S. Suzuki, J. ichi Inoue, and B. K. Chakrabarti, *Quantum Ising Phases and Transitions in Transverse Ising Models* (Springer, Berlin, Heidelberg, 2013).

[39] A. K. Chandra and S. Dasgupta, Floating phase in the one-dimensional transverse axial next-nearest-neighbor ising model, Phys. Rev. E **75**, 021105 (2007).

[40] J. Villain and P. Bak, Two-dimensional ising model with competing interactions: floating phase, walls and dislocations, J. Phys. **42**, 657 (1981).

[41] D. Allen, P. Azaria, and P. Lecheminant, A two-leg quantum ising ladder: A bosonization study of the ANNNI model, J. Phys. A: Math. Gen. **34**, L305 (2001).

[42] H. Rieger and G. Uimin, The one-dimensional ANNNI model in a transverse field: analytic and numerical study of effective hamiltonians, Z. Phys. B **101**, 597 (1996).

[43] P. R. C. Guimarães, J. A. Plascak, F. C. Sá Barreto, and J. Florencio, Quantum phase transitions in the one-dimensional transverse ising model with second-neighbor interactions, Phys. Rev. B **66**, 064413 (2002).

[44] M. Beccaria, M. Campostrini, and A. Feo, Evidence for a floating phase of the transverse annni model at high frustration, Phys. Rev. B **76**, 094410 (2007).

[45] A. Nagy, Exploring phase transitions by finite-entanglement scaling of MPS in the 1d ANNNI model, New J. Phys. **13**, 023015 (2011).

[46] J.-J. Wen, W. Tian, V. O. Garlea, S. M. Koohpayeh, T. M. McQueen, H.-F. Li, J.-Q. Yan, J. A. Rodriguez-Rivera, D. Vaknin, and C. L. Broholm, Disorder from order among anisotropic next-nearest-neighbor ising spin chains in $srho_2o_4$, Phys. Rev. B **91**, 054424 (2015).

[47] C. Karrasch and D. Schuricht, Dynamical phase transitions after quenches in nonintegrable models, Phys. Rev. B **87**, 195104 (2013).

[48] F. Hassler and D. Schuricht, Strongly interacting majorana modes in an array of josephson junctions, New J. Phys. **14**, 125018 (2012).

[49] A. Milsted, L. Seabra, I. C. Fulga, C. W. J. Beenakker, and E. Cobanera, Statistical translation invariance protects a topological insulator from interactions, Phys. Rev. B **92**, 085139 (2015).

[50] I. Peschel and V. J. Emery, Calculation of spin correlations in two-dimensional ising systems from one-dimensional kinetic models, Z. Phys. B **43**, 241 (1981).

[51] T. Giamarchi, *Quantum Physics in One Dimension*, International Series of Monographs on Physics (Clarendon Press, Oxford, 2004).

[52] S. Sachdev, *Quantum Phase Transitions* (Cambridge University Press, Cambridge, 2009).

[53] H. J. Schulz, Critical behavior of commensurate-incommensurate phase transitions in two dimensions, Phys. Rev. B **22**, 5274 (1980).

[54] J. MacQueen, Some methods for classification and analysis of multivariate observations, in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics* (University of California Press, Berkeley, Calif., 1967), pp. 281–297.

[55] J. M. Kosterlitz and D. J. Thouless, Ordering, metastability and phase transitions in two-dimensional systems, J. Phys. C **6**, 1181 (1973).

[56] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, Density-based clustering, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **1**, 231 (2011).

[57] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise* (AAAI Press, Palo Alto, 1996), pp. 226–231.

[58] Y. Bengio, A. Courville, and P. Vincent, Representation learning: A review and new perspectives, IEEE Transactions Pattern Analysis Machine Intelligence **35**, 1798 (2013).

[59] J. MacQueen, Some methods for classification and analysis of multivariate observations, in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics* (University of California Press, Berkeley, Calif., 1967), pp. 281–297.

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Scikit-learn: Machine learning in Python, J. Machine Learning Res. **12**, 2825 (2011).

[61] J. H. Friedman, Greedy function approximation: A gradient boosting machine, Annals of Statistics **29**, 1189 (2000).