Supervised machine learning algorithms based on generalized Gibbs ensembles

Tatjana Puškarov* and Axel Cortés Cubero†
Institute for Theoretical Physics, Center for Extreme Matter and Emergent Phenomena, Utrecht University, Princetonplein 5, 3584 CC Utrecht, the Netherlands
(Dated: October 14, 2019)

Machine learning algorithms often take inspiration from established results and knowledge from statistical physics. A prototypical example is the Boltzmann machine algorithm for supervised learning, which utilizes knowledge of classical thermal partition functions and the Boltzmann distribution. Recent advances in the study of non-equilibrium quantum integrable systems, which never thermalize, have lead to the exploration of a wider class of statistical ensembles. These systems may be described by the so-called generalized Gibbs ensemble, which incorporates a number of "effective temperatures". We propose that these generalized Gibbs ensembles can be successfully applied as the basis of a Boltzmann-machine-like learning algorithm, which operates by learning the optimal values of effective temperatures. We apply our algorithm to the classification of handwritten digits in the MNIST database. While lower error rates can be found with other state-of-the-art algorithms, we find that our algorithm reaches relatively low error rates while learning a much smaller number of parameters than would be needed in a traditional Boltzmann machine, thereby reducing computational cost.

I. INTRODUCTION

Supervised learning consists of approximating a function based on the input-output pairs supplied in a *training* data set. Besides capturing the input-output relationships in the training set, the approximated function should be general enough so that it can be used to map new input examples supplied in a *test* data set. Machine learning algorithms do this by introducing a model with flexible parameters which are learned from the training data set. The learning is done by adjusting the parameters to minimize a "cost function" which measures the distance of the inferred mapping from the actual one.

A simple example of a supervised learning problem is that of the classification of handwritten digits. The Modified National Institute of Standards and Technology (MNIST) database is a large database of handwritten digits used for training and testing image processing models [1]. The data consists of 60000 training and 10000 testing images and their corresponding labels. The input data in this case is a set of grayscale pixels which form the image of a handwritten digit. The images are 28×28 pixels with 256 grayscale values. The output are the corresponding "class" labels of the images, i.e. the correct digits (0-9) that the images represent. Creators of the database trained several types of algorithms and achieved test set error rates ranging from 12% for a linear classifier to 0.8% for a support vector machine [2]. At the moment, an algorithm with a committee of 35 convolutional neural networks achieves the best performance on the test set with an error of 0.23% [3].

In this paper, we develop a model for classifying the MNIST dataset, which is inspired by recent advances in the study of the non-equilibrium statistical physics of quantum many body systems. In particular, our supervised learning model will be based on statistical ensembles that describe integrable systems, which are generalizations of the traditional Gibbs ensemble of equilibrium thermodynamics [4–6].

Our algorithm at this point is not competitive with the high levels of precision of other state-of-the-art approaches [3]. The main advantage of our model instead is its simplicity, given that it manages to achieve reasonably low error rates with a comparatively very small number parameters that need to be fitted.

A common approach to classification is the use of probabilistic modeling. Given some input data $\mathbf{i}$, one computes the probability $P_{\mathbf{i}}^{\mathbf{o}}$ that it corresponds to a particular output class $\mathbf{o}$. Given an input, this will produce a different probability for each possible output class, and the class with the highest probability is selected as the algorithm's prediction. The goal is then to come up with some probability distributions, $P_{\mathbf{i}}^{\mathbf{o}}$, which have flexible parameters that can be learned from a training data set.

The main goals of statistical mechanics are very similar to those of supervised machine learning. In statistical mechanics, one is interested in computing macroscopic observable quantities (such as energy density, pressure, etc.) corresponding to a given configuration of microscopic particles. One is not necessarily interested in the specific

* t.puskarov@uu.nl
† a.cortescubero@uu.nl

individual dynamics of each particle, since slightly different microscopic configurations lead to the same macroscopic output. This line of reasoning parallels the goals of supervised machine learning, where the input variables can correspond to a microscopic configuration of particles, and the output can be viewed as a macroscopic observable.

This conceptual connection to statistical mechanics gave rise to a popular probabilistic model, with many possible applications to machine learning [7–10], the so-called (restricted) Boltzmann machine (RBM/BM). Boltzmann machines are generative models which learn the probability distribution of data using a set of hidden variables. In a typical Boltzmann machine the variables are distributed in two connected layers - a visible and a hidden layer. Since fully connected Boltzmann machines have proved impractical, the connections between the variables are usually constrained - in the case of a restricted Boltzmann machine the only non-vanishing weights are the interlayer ones connecting the visible and the hidden layer, while the intralayer ones are set to zero.

The idea is that the probabilities $p(v^{\mathbf{i}}, h)$ of configurations $(v^{\mathbf{i}}, h)$ can be computed, as is done in statistical physics, from a Boltzmann distribution, given by

$$p(v^{\mathbf{i}}, h) = \frac{\mathrm{e}^{-\beta E(v^{\mathbf{i}},h)}}{Z}, \qquad Z = \sum_{v^{\mathbf{i}},h} \mathrm{e}^{-\beta E(v^{\mathbf{i}},h)}, \tag{1}$$

where $v^{\mathbf{i}}$ are the visible units which are fed the input data $\mathbf{i}$ and h are the hidden units. The parameter β corresponds to the inverse temperature and is traditionally set to $\beta = 1$, although there have recently been explorations of temperature-based BM [11, 12], and $E(v^{\mathbf{i}}, h)$ can be thought of as the energy of a given configuration. One useful form of the energy function that can be used is that corresponding to an Ising-like model [13–15],

$$E(v^{\mathbf{i}}, h) = \sum_i b_i^h h_i + \sum_i b_i^{\mathbf{i}} v_i^{\mathbf{i}} + \sum_{i,j} w_{i,j}^{\mathbf{i}}\, h_i v_j^{\mathbf{i}}, \tag{2}$$

where h_i and $v_j^{\mathbf{i}}$ are stochastic binary units which represent the hidden and the input variables, and biases b_i and weights $w_{i,j}$ are free parameters that need to be learned. The goal of the algorithm is to learn, from the training data set, the optimal values of b_i and $w_{i,j}$ that produce the most accurate probabilities $p(v^{\mathbf{i}}, h)$. This is done by setting the states of the visible layer to a training vector and then using those to set the nodes of the hidden layer to 1 with the probability $p = 1/[1 + \exp(-\beta \Delta E_i)]$, where $\Delta E_i = E(h_i = 0) - E(h_i = 1)$. The training of a BM consists in "reconstructing" the visible and hidden vector in this way from one another while adjusting the weights and biases. The weights and biases are adjusted in a procedure called contrastive divergence [16] so that the reconstructed visible vector gets closer to the input one and the energy of the total configuration is decreased, thus increasing the probability $p(v^{\mathbf{i}}, h)$.

The Boltzmann machine is a generative model since it learns a probability distribution which represents the training data and can, after training, generate new data with the learned distribution. It can also be used for classification of data [17], usually as a feature extractor. While training the RBM, the network learns a different representation h of the input data $v^{\mathbf{i}}$, which can be fed to a classification algorithm [7]. In this way, the RBM is used to extract the features h, which can then be fed to, e.g., a single softmax output layer to classify the data. Alternatively, an RBM with a part of its visible layer fed the output data $v^{\mathbf{o}}$ can be used to generate the joint probabilities of the inputs and the output labels. The probability of such a configuration $(v^{\mathbf{i}}, h, v^{\mathbf{o}})$ is

$$p(v^{\mathbf{i}}, h, v^{\mathbf{o}}) = \frac{\mathrm{e}^{-\beta E(v^{\mathbf{i}},h,v^{\mathbf{o}})}}{Z}, \qquad Z = \sum_{v^{\mathbf{i}},h,v^{\mathbf{o}}} \mathrm{e}^{-\beta E(v^{\mathbf{i}},h,v^{\mathbf{o}})}, \tag{3}$$

where the energy is now

$$E(v^{\mathbf{i}}, h, v^{\mathbf{o}}) = \sum_i b_i^h h_i + \sum_i b_i^{\mathbf{i}} v_i^{\mathbf{i}} + \sum_i b_i^{\mathbf{o}} v_i^{\mathbf{o}} + \sum_{i,j} w_{i,j}^{\mathbf{i}}\, h_i v_j^{\mathbf{i}} + \sum_{i,j} w_{i,j}^{\mathbf{o}}\, h_i v_j^{\mathbf{o}}. \tag{4}$$

The RBM then models the joint probability distribution of the input and the output data by summing over the possible states of the hidden units

$$P_{\mathbf{i}}^{\mathbf{o}} = \sum_h p(v^{\mathbf{i}}, h, v^{\mathbf{o}}). \tag{5}$$

Drawing on the success of learning algorithms based on classical statistical mechanics, the natural generalization to *quantum* Boltzmann machines was recently proposed [18]. In this case the energy function is promoted to a Hamiltonian operator, and the data is represented as a particular quantum state. A quantum Ising Hamiltonian can be defined by promoting the classical spin variables z_i to spin operators, which can be expressed in the basis of the Pauli matrices $\sigma_i^{x/y/z}$. One example considered in [18] is the Hamiltonian corresponding to an inhomogeneous Ising chain in a transverse and a longitudinal magnetic field

$$H^{\mathbf{o}} = -\sum_i \Gamma_i \, \sigma_i^x - \sum_i b_i \, \sigma_i^z - \sum_{i,j} w_{i,j} \, \sigma_i^z \sigma_j^z . \tag{6}$$

An input sample $\mathbf{i}$ is represented as a particular state $|z(\mathbf{i})\rangle$, which corresponds to a configuration of spins in the quantum Ising chain. The probabilities can now be written in terms of the quantum Boltzmann distribution, based on the canonical Gibbs ensemble, [19]

$$P^{\mathbf{o}}_{\mathbf{i}} = \frac{\langle z(\mathbf{i})| \, \mathrm{e}^{-\beta H^{\mathbf{o}}} \, |z(\mathbf{i})\rangle}{Z}, \qquad Z = \mathrm{Tr}\left\{ \mathrm{e}^{-\beta H^{\mathbf{o}}} \right\}, \tag{7}$$

where the trace is over all the possible configurations of the input states. In the case of a quantum Boltzmann machine the goal of the algorithm is to learn the different coupling constants of the Hamiltonian $H^{\mathbf{o}}$, such that an input state $|z(\mathbf{i})\rangle$ corresponding to the output $\mathbf{o}$ is the ground state, or close to the ground state of the Hamiltonian $H^{\mathbf{o}}$. In other words, one needs to find the inhomogeneous quantum Hamiltonian whose low-lying eigenstates are close to $|z(\mathbf{i})\rangle$.

In the next section we will introduce our new probabilistic model based on a generalization of the Gibbs ensemble, which has recently become prominent in the study of integrable quantum many-body systems out of equilibrium. This generalized Gibbs ensemble (GGE), incorporates a large number of quantities known as "effective temperatures", which are generalizations of the parameter, β. We will expose in some detail the structure of the GGE for a simple quantum Ising chain, which will be the basis of our model. We will show in Section III how these effective temperatures can be treated as the adjustable parameters to be learned from the training data set, yielding a computationally cheap and reasonably effective learning algorithm. We apply our GGE-based model in the problem of classification of MNIST hand-written digits, and compare our results to previous established classification algorithms.

II. THE GGE MACHINE

The main goal of this paper is to explore the utility of applying different physics-inspired probability distributions to supervised learning, and to see if they can have any advantage over the (quantum) Boltzmann distributions.

In recent years, there has been significant progress in our understanding the dynamics of quantum many-body systems out of thermal equilibrium, where concepts such as the quantum Boltzmann distribution are not applicable. In particular, there has been a large amount of work towards understanding the non-equilibrium dynamics of one-dimensional integrable quantum systems (for a review, see [20–22]).

Integrable quantum systems are characterized by having a large number of conserved quantities (whose expectation value does not change under time evolution). We can write these conserved quantities as quantum operators Q_n, labeled by some integer n, with the property that they commute with the Hamiltonian, $[H, Q_n] = HQ_n - Q_nH = 0$. This large number of dynamical constraints usually results in these systems being exactly solvable, and enables analytic computation of certain quantities [23, and references therein]. One important result in the study of integrable systems out of equilibrium, is the realization that even after very long times, these systems never reach a state of thermal equilibrium. It is now understood that at long times, physical observables of integrable quantum systems generally reach an equilibrium state described by a generalized Gibbs ensemble (GGE) [4, 5, 24], where the probabilities associated with a given state $|\Psi\rangle$ are given by

$$P_\Psi = \frac{\langle \Psi| \, \mathrm{e}^{-\sum_n \beta_n Q_n} \, |\Psi\rangle}{Z}, \qquad Z = \mathrm{Tr}\left\{ \mathrm{e}^{-\sum_n \beta_n Q_n} \right\}. \tag{8}$$

where β_n can be considered to be "effective temperatures" corresponding to each of the higher conserved quantities of the integrable model. It is important to point out that in quantum integrable models, the conserved quantities, Q_n, are generally extensive and can be expressed as the sum of local operators, which are essential properties needed for (8) to be a reasonable ensemble for statistical physics [25].

In this paper we will implement a supervised learning algorithm to analyse the MNIST handwritten digit data, based on the generalized Gibbs ensemble (8), instead of the traditional Boltzmann ensemble.

The main appeal of using a GGE in a machine learning algorithm is that it may be possible to store a large amount of non-trivial information in the effective temperature parameters, β_n. The effective temperature variables contain information directly related to the macroscopic quantities of a system. It can then be reasonably expected that if one learns a handful of effective temperatures corresponding to a given macroscopic output, this may carry more essential information than learning a similar number of microscopic parameters, such as the coupling between two particular input variables. We then propose that in some cases, it should be more useful to learn a set of effective temperatures, β_n, than to learn the full set of microscopic couplings, Γ_i, b_i, $w_{i,j}$. This hypothesis is further motivated by the results of [11], which shows how the temperature is a very useful parameter to learn in a traditional Boltzmann machine; in our case, this idea is exploited by having a large number of temperature-like parameters.

We test this idea on the MNIST data set. We assume the system is described by a simple, homogeneous and integrable Hamiltonian similar to (6) where we can set $\Gamma_i = \Gamma$, $w_{i,i+1} = w$, $b_i = 0$. This is the prototypical transverse-field Ising (TFI) model

$$H = -w \sum_{i=1}^{L} \sigma_i^z \sigma_{i+1}^z + \Gamma \sum_{i=1}^{L} \sigma_i^x. \tag{9}$$

The Hamiltonian (9) describes an integrable spin chain which can be diagonalized by a Jordan-Wigner and a Bogoliubov transformation [15] to give a free fermion model

$$H = \sum_k \varepsilon_k \left(\eta_k^\dagger \eta_k - \frac{1}{2} \right), \tag{10}$$

where the single particle energies are $\varepsilon_k = \sqrt{w - 2\Gamma \cos\theta_k + \Gamma^2}$ and $\mathrm{e}^{\mathrm{i}\theta_k} = (\Gamma - \mathrm{e}^{\mathrm{i}k})/\sqrt{1 + \Gamma^2 - 2\Gamma\cos k}$ [5]. The momenta are quantized as $k_j^{\mathrm{Even}} = \frac{2\pi}{L}(j + \frac{1}{2})$ in the even and $k_n^{\mathrm{Odd}} = \frac{2\pi j}{L}$ in the odd sector, and $j = -\frac{L}{2}, \ldots, \frac{L}{2} - 1$. The even or odd sector of momentum values correspond to whether periodic or antiperiodic boundary conditions are imposed on the free fermion operator. We interpret the input data, vectors $|\mathbf{i}\rangle$ of 256 grayscale values for each pixel, as the eigenstates of the Hamiltonian (9). In order to do that, we binarize the MNIST data by setting a pixel value to 0 if it is smaller than 256/2, and 1 otherwise. The states are then given in the basis of the occupation number operator of the Bogoliubov fermions η_k - a 0 pixel means that the corresponding fermionic excitation is not occupied and a pixel 1 that it is occupied. A state belongs to the even/odd sector if it has a total even/odd number of excitations. From a practical standpoint, binarizing the data is not necessary, but we do it here to keep in line with the physical interpretation, given that fermions satisfy the Pauli exclusion principle.

The learning part of the algorithm consists in optimizing the set of effective temperatures that reproduce the output data $\mathbf{o}$. Since the input data represents a certain state and the Hamiltonian parameters are preset, the dependence on the output is shifted to the effective temperatures, $\beta_n^{\mathbf{o}}$. The probability that a configuration of pixels $|\mathbf{i}\rangle$ represents a digit $\mathbf{o}$ is

$$P_{\mathbf{i}}^{\mathbf{o}} = \frac{\langle \mathbf{i}|\, \mathrm{e}^{-\sum_n \beta_n^{\mathbf{o}} Q_n} \,|\mathbf{i}\rangle}{Z} \tag{11}$$

where Q_n are the conserved charges of the TFI chain. In terms of the Bogoliubov fermions, they have the simple form [6]

$$\begin{aligned} Q_n^+ &= \sum_k \varepsilon_k \cos\left[nk\right] \eta_k^\dagger \eta_k, \\ Q_n^- &= -2w \sum_k \sin\left[(n+1)k\right] \eta_k^\dagger \eta_k. \end{aligned} \tag{12}$$

The even charges Q_n^+ are defined for $n = 0, 1, 2, \ldots, L-2$, whereas the odd charges Q_n^- are defined for $n = 1, 2, \ldots, L-2$.The training of the algorithm corresponds to learning sets of Lagrange multipliers $\beta_n^{\mathbf{o}}$ for each digit $\mathbf{o}$ in order to produce the appropriate probabilities in (11), with the further simplification that we only learn the conditional probabilities

$$P_{\mathbf{i}}^{\mathbf{o}} = \frac{\langle \mathbf{i}|\, \mathrm{e}^{-\sum_n \beta_n^{\mathbf{o}} Q_n} \,|\mathbf{i}\rangle}{\sum_{\mathbf{o}=0}^{9} \langle \mathbf{i}|\, \mathrm{e}^{-\sum_n \beta_n^{\mathbf{o}} Q_n} \,|\mathbf{i}\rangle}, \tag{13}$$

thus circumventing the difficulty of calculating the full partition function. As is standard practice in neural networks, the algorithm additionally learns "biases". These are Lagrange multipliers $\beta_{\mathbb{1}}^{\mathbf{o}}$ which corresponds to a trivial charge, the identity $Q_{\mathbb{1}} = 1$.

The general expectation is that one can reach a reasonable level of accuracy in classifying the images by including a subset of the conserved charges Q_n and learning the corresponding parameters $\beta_n^{\mathbf{o}}$. On the other hand, the traditional

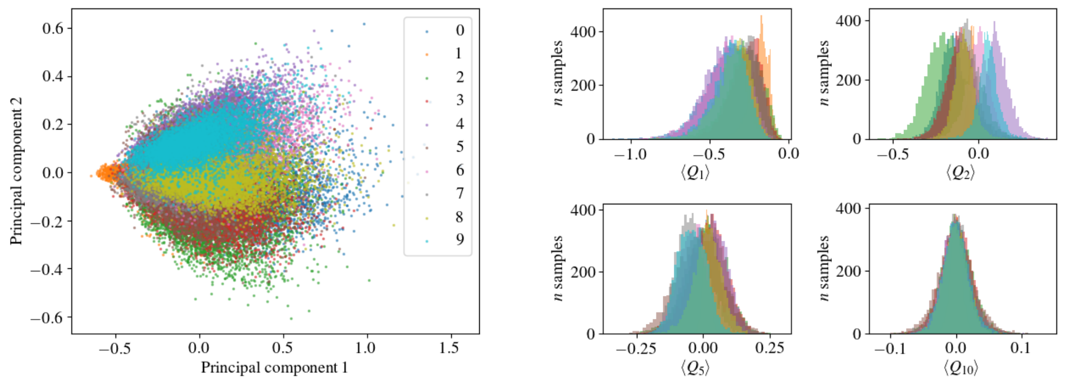

FIG. 1. Left: PCA projection onto the two first principal components of $N = 21$ conserved charges calculated from the MNIST training data. PCA finds a lower-dimensional representation of the data by using an orthogonal transformation to project the data onto uncorrelated directions. Right: Histograms of the expectation values of several charges (Q_1, Q_2, Q_5 and Q_{10}) for all the training samples. In both cases, the charges were calculated using the Hamiltonian parameters $w = 1.0$, $\Gamma = 1.0$. The samples are colored according to the digit they represent.

Boltzmann machine would need to learn a larger set of parameters in order to reach the same accuracy. The GGE-based algorithm can then provide a computationally much cheaper alternative. We analyze and support this in the following section.

In the GGE-based algorithm the goal is to start with a very simple Hamiltonian, but instead learn the variables that produce a non-trivial ensemble. One then learns the effective temperatures such that the state $|\mathbf{i}\rangle$ is a typical state occurring with high probability in the given ensemble. This is different to the interpretation of the quantum Boltzmann machine, where one finds a generally complicated and non-trivial Hamiltonian, such that $|\mathbf{i}\rangle$ is close to its ground state.

III. THE ALGORITHM AND PERFORMANCE ON THE MNIST DATASET

The MNIST data is supplied as $(784, 1)$ vectors, read from the 28×28 images line by line starting from the top left corner and binarized. From it, we extract a total of N conserved charges which are the features we later feed to a neural network. For a given N, we calculate the expectation values of the same number of even Q_n^+ and odd charges Q_n^- defined in eq. (12) if N is even, or one extra even charge if N is odd. The features we select are therefore

$$Q_n = \begin{cases} Q^+_{n/2} & \text{if } n \text{ even} \\ Q^-_{(n+1)/2} & \text{if } n \text{ odd} \end{cases}, \quad n = 0, 1, 2, \ldots, N-1. \tag{14}$$

Principal component analysis (PCA) plot and histograms in fig. 1 show clustering of the data for each digit indicating that the conserved charges for a particular digit do capture the similarities between different training instances of that digit. On the other hand, there are significant overlaps between the different clusters, which is expected considering that we discard a lot of information by using only N charges. We use PCA only to illustrate the data, and not to change the basis before feeding the data to the network.

Having selected the features Q_n with $n = 0, 1, \ldots, N-1$, we train a fully connected single-layer neural network. The input layer x consists of the N selected features which are rescaled to have zero mean and unit variance, as is standard practice in neural networks [26]. The input layer is connected to a softmax output layer y with 10 nodes, corresponding to the ten possible digits. There is an additional bias node that connects to each output node. The weights connecting the layers are contained in the matrix $W_{10\times N}$ and the bias weights in the vector b. The output is calculated as

$$y = \phi(Wx + b), \tag{15}$$

where ϕ is the activation function, in this case the softmax function [17]. This corresponds to eq. (13) where, for each training sample, x is a vector of the N conserved charges Q_n and y is a vector containing the output probabilities $P^{\mathbf{o}}$ for all the digits $\mathbf{o}$. The weights matrix W contains the Lagrange multipliers $\beta_n^{\mathbf{o}}$, and the biases are the multipliers corresponding to the identity charge $\beta_{\mathbb{1}}^{\mathbf{o}}$.

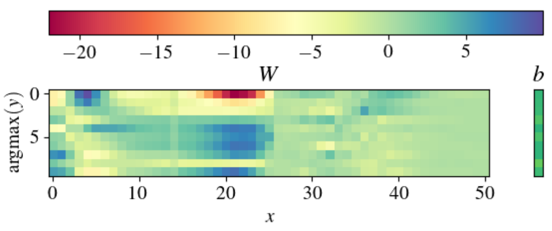

FIG. 2. Typical weights W and biases b of eq. (15) after training a neural network described in the text. In this case, the network is fed $N = 51$ conserved charges calculated from the original MNIST training data using $w = 1$, $\Gamma = 1$.

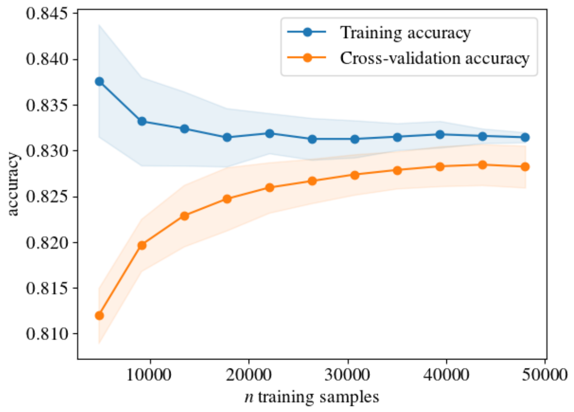

FIG. 3. Learning curves for the model with the input layer size $N = 51$ and the charges calculated with $w = 1$, $\Gamma = 2$. Learning curves are the training- and cross-validation accuracies with respect to the number of instances used for training.

The weights and biases are initialized at random values and they are learned by training the network to minimize the cost function, which is the cross-entropy given by

$$S(y,t) = -\sum_{i=1}^{M} t \log y, \tag{16}$$

including an L_2 regularization term [17, 27]. Here t are the target values of the output and M is the number of samples in the batch. The cost function measures the distance of our predictions y to the target output values t and by tuning the parameters W and b we try to minimize that distance for all the M training samples. The minimization is done using the limited-memory BFGS algorithm [28].

The accuracy rates depend on a number of hyperparameters (parameters of the algorithm and not the model). These include the regularization parameter, the learning rate, the stopping criterion and the maximum number of epochs [17, 27]. The optimal values of the hyperparameters are chosen using randomized parameter optimization [29] and stratified 10-fold cross-validation. In this procedure the full training set is split into three parts called "folds" (with sample distribution in each fold similar to the full dataset distribution), the model is trained on nine of the folds, and the performance is measured on the remaining, previously unseen, one. This is repeated ten times so that each fold acts as the validation set once and the performance is averaged. The procedure is repeated for each combination of the hyperparameters. The hyperparameter values for the combinations are picked at random from a specified range (for continuous parameters) or a set (for discrete parameters).

With the selected hyperparamters the network is trained on the $M = 60000$ samples of the training set using stochastic average gradient descent and stratified 10-fold cross-validation. Typical weights and biases learned are shown in fig. 2, whereas fig. 3 shows typical learning curves. We can conclude that the model is not overfitted, but the relatively high error might indicate a high bias with respect to the data. However, since we significantly truncate the number of input features thus losing a portion of the information, the high error is not unexpected. Furthermore, since the two curves converge, adding additional training instances would most likely not improve the performance.

The algorithm performance is measured by the accuracy of the prediction on the $M_{\text{test}} = 10000$ samples of the test set. We plot the observed accuracies with respect to the number of charges N kept in the truncated GGE and with respect to the Hamiltonian parameter Γ in fig. 4. In this section, $J = 1$. As is expected, keeping a larger number of charges in the GGE improves performance. However, the accuracy rate saturates at about 90% with $N \approxeq 80$ and does not improve by further adding of the charges to the ensemble - the accuracy rate when using $N = 784$ charges with $\Gamma = 1.0$ is still 90.9%. While this score is lower than what state of the art algorithms achieve, it nevertheless gives a systematic way of reducing the number of parameters of the model. The simplest neural network, trained in the original MNIST paper [2], consists of a single layer of 784 input nodes which are fed the pixel data and are connected to the 10 output nodes. This model has $n(\text{params}) = n(\text{in nodes}) \times n(\text{out nodes}) + n(\text{bias nodes}) = 7850$ free parameters and yields 88.0% accuracy. This is similar to the GGE model with $N = 784$ charges at $\Gamma = 1.0$ and 7850 free parameters which yields 90.9% accuracy, a slight improvement as opposed to using the raw pixel data.

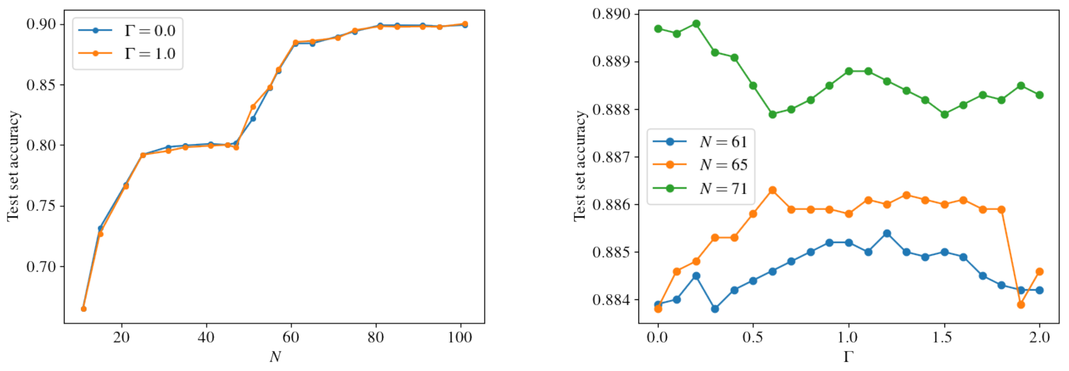

FIG. 4. The test set accuracy as a function of the size of the input layer N and the Hamiltonian parameter Γ.

However, we can reach similar accuracies even with a much smaller number of parameters, as exemplified by fig. 4. A network with $N = 91$ and $\Gamma = 2.0$ has 920 parameters and yields 89.8% accuracy, whereas a network with $N = 61$ and $\Gamma = 1.0$ has 620 parameters and yields 88.5%.

For further comparison, we have trained a standard classification restricted Boltzmann machine (as defined in the introduction) [7] to sort the MNIST data. In this case, the network consists of a visible layer of 784 input stochastic binary nodes, a hidden layer with n(hidden nodes) stochastic binary nodes and an output layer with 10 binary nodes. The input, raw MNIST data rescaled to the interval $[0, 1]$, is fed to the visible layer. The nodes of the hidden layer are connected to the visible nodes and are activated by a sigmoid function. The energy of this system is given in eq. (2), and the weights and biases are learned using the contrastive divergence algorithm such that the energy is minimized. Using a grid search for the hyperparameters, we find that the best configuration is a network with 100 hidden nodes, thus learning $n(v^{\mathbf{i}}) \times n(h) + n(v^{\mathbf{i}}) + n(h) = 79284$ parameters. The hidden nodes are further connected to a softmax output layer, as in the GGE case, and this part of the network is trained as a supervised model. This adds an additional $n(v^{\mathbf{o}}) \times n(h) + n(v^{\mathbf{o}}) = 1010$ parameters, for a total of $n(\text{parameter}) = 80294$. The accuracy on the test set is in this case 96.1%. To reiterate, in the described setup, RBM is used to extract 100 features which are used for classification. Using 100 GGE conserved quantities in contrast yields test set accuracy of 90.0%. This is not surprising since the features which the RBM learns have no restrictions in terms of analytic forms, which is the case with the GGE conserved quantities which are physical quantities. In order to compare the computational complexities of the two approaches, we train an RBM with $n(\text{hidden nodes}) = 45$ and optimized hyperparameters, which achieves a similar performance to our GGE algorithm. In this example, the RBM has a test set accuracy of 89.4% with the trade off of learning a total of 36569 parameters, while the GGE algorithm achieves 90.0% with learning 1020 parameters with $N = 101$ and $\Gamma = 1.0$. While there is a computational cost associated to calculating the charges we feed to the network, this is still a decrease in the total cost. The key difference here is the fact that the GGE algorithm assumes a simple Hamiltonian (9) with homogeneous coupling, whereas the RBM learns an inhomogeneous Hamiltonian with many different coupling constants.

IV. CONCLUSIONS

Inspired by the parallels between statistical mechanics and supervised learning, we described a machine learning algorithm based on the generalized Gibbs ensemble. The algorithm assumes that the input is an eigenstate of a simple Hamiltonian, and uses this to extract conserved charges from the inputs. Keeping only a small number of those, the algorithm then learns the effective temperatures such that the state is a typical state occurring with high probability in the given ensemble.

The effective temperatures of the GGE are directly related to macroscopic observable quantities. It is therefore expected that these parameters are more efficient quantities to learn than the typical couplings learned in a Boltzmann machine.

The biggest advantage of our new GGE algorithm is that it seems to achieve reasonably low error rates, while learning a comparatively low number of parameters. This advantage was shown explicitly by comparing with a restricted Boltzmann machine, where it was shown that the GGE algorithm outperforms a RBM with around 36 times the number of learned parameters.

While our GGE algorithm currently does not outperform state-of-the art learning algorithms in terms of low error rates, it still proves to be a useful way to reduce the number of parameters to be learned. It is expected the GGE algorithm can be further improved in the future, perhaps by adding more hidden layers, making it a more competitive alternative.

ACKNOWLEDGMENTS

The authors would like to thank Dirk Schuricht for valuable discussions and for proof reading the manuscript. ACC's work is supported by the European Union's Horizon 2020 under the Marie Sklodowoska-Curie grant agreement 750092. This work is part of the D-ITP consortium, a program of the Netherlands Organisation for Scientific Research (NWO) that is funded by the Dutch Ministry of Education, Culture and Science (OCW).

[1] Available at http://yann.lecun.com/exdb/mnist/.
[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Proceedings of the IEEE **86**, 2278 (1998).
[3] D. Ciregan, U. Meier, and J. Schmidhuber, in *Computer vision and pattern recognition (CVPR), 2012 IEEE conference on* (IEEE, 2012) pp. 3642–3649.
[4] M. Rigol, V. Dunjko, V. Yurovsky, and M. Olshanii, Physical review letters **98**, 050405 (2007).
[5] P. Calabrese, F. H. Essler, and M. Fagotti, Journal of Statistical Mechanics: Theory and Experiment **2012**, P07022 (2012).
[6] M. Fagotti and F. H. Essler, Physical Review B **87**, 245107 (2013).
[7] G. E. Hinton, S. Osindero, and Y.-W. Teh, Neural computation **18**, 1527 (2006).
[8] G. E. Hinton and R. R. Salakhutdinov, science **313**, 504 (2006).
[9] H. Larochelle and Y. Bengio, in *Proceedings of the 25th international conference on Machine learning* (ACM, 2008) pp. 536–543.
[10] A. Coates, A. Ng, and H. Lee, in *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (2011) pp. 215–223.
[11] G. Li, L. Deng, Y. Xu, C. Wen, W. Wang, J. Pei, and L. Shi, Scientific reports **6**, 19133 (2016).
[12] L. A. Passos and J. P. Papa, Neural Processing Letters , 1 (2017).
[13] J. J. Hopfield, Proceedings of the national academy of sciences **79**, 2554 (1982).
[14] G. E. Hinton and T. J. Sejnowski, in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (Citeseer, 1983) pp. 448–453.
[15] E. Lieb, T. Schultz, and D. Mattis, Annals of Physics **16**, 407 (1961).
[16] G. E. Hinton, in *Neural networks: Tricks of the trade* (Springer, 2012) pp. 599–619.
[17] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, Vol. 1 (MIT press Cambridge, 2016).
[18] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, arXiv preprint arXiv:1601.02036 (2016).
[19] There are slight modifications to this expression in the case where there are hidden variables, which can be found in [18].
[20] A. Polkovnikov, K. Sengupta, A. Silva, and M. Vengalattore, Reviews of Modern Physics **83**, 863 (2011).
[21] J. Eisert, M. Friesdorf, and C. Gogolin, Nature Physics **11**, 124 (2015).
[22] C. Gogolin and J. Eisert, Reports on Progress in Physics **79**, 056001 (2016).
[23] G. Mussardo, *Statistical field theory: an introduction to exactly solved models in statistical physics* (Oxford University Press, 2010).
[24] T. Barthel and U. Schollwöck, Physical review letters **100**, 100601 (2008).
[25] L. Vidmar and M. Rigol, Journal of Statistical Mechanics: Theory and Experiment **2016**, 064007 (2016).
[26] Not rescaling the input data, which is better suited to the physical analogy that we use, yields approximately 1% lower validation scores as compared to the same network architecture fed with rescaled data. For example, a network with $N = 51$ and $\Gamma = 1.0$ yields 83.2(3)% accuracy with rescaled and 81.5(4)% with raw, not scaled, charges. For the comparison, each networks' hyperparameters were optimized using a random search.
[27] 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, Journal of Machine Learning Research **12**, 2825 (2011).
[28] D. C. Liu and J. Nocedal, Mathematical programming **45**, 503 (1989).
[29] J. Bergstra and Y. Bengio, Journal of Machine Learning Research **13**, 281 (2012).