NANOSYSTEMS:
PHYSICS, CHEMISTRY, MATHEMATICS

Lobanov I.S. *Nanosystems:*
Phys. Chem. Math., 2022, **13** (6), 593–607.
http://nanojournal.ifmo.ru
DOI 10.17586/2220-8054-2022-13-6-593-607

Original article

Spin Boltzmann machine

Igor S. Lobanov

Faculty of Physics, ITMO University, Saint Petersburg, Russia

igor.lobanov@metalab.ifmo.ru, lobanov.igor@gmail.com

PACS 75.78.-n, 05.65.+b

ABSTRACT Boltzmann machine (BM) is a recurrent network, which has a wide range of applications in machine learning (ML) including dimensionality reduction, feature learning and classification. Standard BM is described by the Ising model and can be implemented as a spin ice based device. Such hardware implementation is faster and more energy efficient than a simulation on digital computers. At the moment, a hardware BM is a single purpose device designed on digital computers for a specific task. In the paper we propose a generalized BM capable of fitting parameters by demonstration of training examples, which is done completely inside the spintronic device. Our generalization is based on the Heisenberg model, which is more accurate than the Ising model for spin ice. We show that for some systems minimization of Kullback-Leibler divergence during training of BM is equivalent to minimization of free energy with respect to the biases of the units, hence training of the ML model can be done by energy dissipation. We include the biases as degrees of freedom of the device, whose dynamics is described by the same Landau-Lifschitz-Gilbert equation as for spins representing units of BM. The demonstration of samples from the training set is done by fixing inputs and outputs according to ground truth. The training samples are remembered by the machine becoming minima on the energy landscape implementing a kind of long-term potentiation. The performance of the proposed machine is compared with a single layer perceptron artificial neural network and with a Bernoulli restricted BM on a binary classification problem.

KEYWORDS spectral gap, quantum graph, Schrödinger operator, discrete spectrum.

ACKNOWLEDGEMENTS The work is supported by Russian Science Foundation grant 22-22-00565:
https://rscf.ru/en/project/22-22-00565/.

FOR CITATION Lobanov I.S. Spin Boltzmann machine. *Nanosystems: Phys. Chem. Math.*, 2022, **13** (6), 593–607.

1. Introduction

Advances in machine learning (ML) lead to a breakthrough in the field of artificial intelligence in recent decades. We have seen great progress in previously intractable problems, such as speech recognition, natural language translation, computer vision, motion planning [1] and even in physics itself [2]. The main approaches to ML were inspired by biology, e.g. artificial neural networks (ANN), or physics, e.g. Boltzmann machine (BM). The basic theory of ANN was formulated in the 1940s [3], but soon the research stagnated due to lack of processing power [4]. The rate of development was accelerated dramatically with the appearance of graphical processing units (GPU) offering huge processing power to solve linear algebra problems for a relatively small price. At the moment computers can solve many artificial intelligence (AI) tasks, however the energy efficiency of the organic brain is still much higher. The need to reduce energy consumption leads to the emergence of dedicated AI accelerators [5]. The widespread digital computers are good for precise computations, but analog devices and stochastic computing better suit the needs of ML [6]. For example, linear systems can be solved in constant time by analog computers, which offers a huge speed boost [7].

At the moment, there are many teams trying different approaches to improving ML hardware. It was shown in [8] that almost arbitrary wave dynamics can be considered as a variant of recurrent neural networks. Spintronics devices are one of promising hardware platforms for biologically inspired computing [9]. In [10] nanoscale spintronic oscillators were used to implement the basic block of ANN combining nonlinearity and memory, a successful recognition of spoken digits is reported. The inverse Rashba-Edelstein magnetoelectric neuron was proposed in [11], its effectiveness was demonstrated on handwritten images recognition. $784\times200\times10$ deep belief network consisting of p-bit-based neurons was created using spin–orbit-torque magnetic random access memory in [12]. Magnetic skyrmions, stabilized due to topological protection, are prospective novel information carriers [13], which is used in multiple nonclassic computing devices [14, 15]. Artificial synapse was created using magnetic skyrmion in [16]. Spintronic artificial synapse was controlled by ultrashort laser pulses in [17]. Random skyrmion textures were used for reservoir computing [18].

Most approaches to ML distinguish training phase (parameter fitting) and prediction phase. Emerging devices for neuromorphic computing based on new physical principles are devoted to the prediction phase. The prediction is a crucial part of usage of ML algorithms, namely this part is implemented in consumer products, which can benefit from lower

energy usage or faster response of novel devices. However, model training is much more time and energy consuming than prediction. Besides repeated computation of the model, the training includes additional steps such as computation of loss, differentiation of the loss function with respect to the parameters, inference of hidden variables and so on. These actions are not trivially connected with model computation and require additional hardware support. At the current stage of research, the training is rarely made on the device itself, instead it is common to fit the parameters by simulation of the device in a digital computer. In the article we are going to go further and suggest a quite general approach for ML model training directly in the device without digital computers.

The training phase consists in fitting parameters by minimizing loss function (or maximizing likelihood). Energy in dissipative systems naturally decreases, hence an appealing idea is to map the loss function to system energy, thus obtaining automatic training. Quasi-thermodynamics and free energy minimization have been discussed in the context of ML for several decades. Hopfield network [19–21] and BM [22–24] minimize free energy thus converging to memorized states. The energy minimization in BM is used for prediction, training is carried out by a separated procedure, and we are going to unify the processes in our work. In [6] dissipative systems are noted to be popular for simulation of associative memory. In [25] the neuron activity was recognized to be described by pseudo-thermodynamics and the training can be simulated by annealing. In [26] a dissipative quantum model of the brain was shown to match the formation of coherent domains of synchronized neuronal oscillatory activity and phase transitions obtained from neurophysiological data collected from electroencephalograms. In [27] it was observed that the training of ANN can be considered as a dissipative process. In the article [28] an ANN was trained to model energy dissipation, which provides advantages in the reconstruction of magnetic resonance and computed tomography images. In [29] artificial spintronic synapses were designed and shown to be suitable for Hopfield model-based associative memory.

The reversible computing [30] is a topic closely related to ML, where energy minimization is used as the driving force of computation. Quantum mechanics tells us that all computations are reversible, and this is indeed true for quantum gates [31]. The classical computers are not reversible, in particular, due to the loss of information, which leads to an increase in ambient temperature. Elementary steps of classic computation are usually defined as logic gates, which are mappings from inputs to outputs. The symmetry of roles of inputs and outputs can be achieved, if both inputs and outputs are defined by an implicit relation between them. The reversibility in classical computing can be restored, if assumptions of the inverse function theorem are applicable to the relation. There are several physical implementations of reversible computations. In applications it is convenient to define an error function, which represents mismatch between inputs and outputs. If the error can be equated to energy of a physical system, the agreement between inputs and outputs can be achieved by dissipation. The reversible computing devices were created using electrical circuit elements with memory [32]. Invertible boolean logic with BM was described and partially implemented in CMOS-assisted nanomagnet-based hardware in [33]. A magnetic tunnel junctions based device was created in [34], which was able to factorize several numbers (945 is the largest one) using reversible computing in the form of BM. As was demonstrated in [35] by simulation, the digital memcomputing machine is able to solve the boolean satisfiability problem in polynomial time. Reversible elements can be created from nanomagnets as was demonstrated in [36]. Larger magnets permit spin waves, which also can be used for implementation of reversible magnetic logic gates [37]. Large magnetic systems may contain topologically protected solitons, such as skyrmions, which can be used as information carriers for large-scale reversible computation [38].

Lot of new physical devices for ML are variants of BM. Standard BM is described by the Ising model. BM can be naturally implemented in spin-ice [39]. Artificial spin-ice (ASI) systems are constructed from arrays of dipolar coupled monodomain magnets [40, 41]. At the moment ASI is used to embody novel functional devices for convenient logic and for neuromorphics [42]. ASI has many metastable-states, they are configured by an external field and are able to store and process temporal input patterns [43]. Thermally driven balanced NAND gate was implemented in kagome spin ice in [44]. Fabrication of computational blocks in spin ice was demonstrated by an experiment in [45].

In the paper we consider a variant of BM based on the Heisenberg model, where orientation of the magnetic moments is not constrained, but can be controlled by anisotropy. We augment the model with additional spins, which generate an effective field playing role of biases in standard BM, thus obtaining a spin Boltzmann machine (SBM). Just like BM, SBM can be implemented as a spin-ice device. The energy dissipation is used for training of SBM, and we describe two schemes of training with spatial or temporal separation of samples. SBM does not rely on external devices for training, it can be retrained on another training set to solve another problem, making SBM reusable.

The paper is organized as follows. In Section 2 we show how minimization of Kullback–Leibler divergence for Boltzman distribution can be reduced to minimization of energy. In Section 3 we recall the definition and principles of operation of BM. In Section 4 we formulate a binary classification problem and solve the problem using ANN and BM. In Section 5 we introduce SBM and describe how BM can be made flexible by introduction of additional degrees of freedom storing biases. In Section 6 we apply SBM to solve of the binary classification problem and compare the result with the result of ANN and BM.

2. Relaxation as distribution learning

Consider problem of estimation of parameters of multivariate normal distribution:

$$\rho(\boldsymbol{x}|\Sigma,\boldsymbol{\mu}) = \frac{1}{Z}\exp(-E[\boldsymbol{x}]), \quad E[\boldsymbol{x}] = \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})\cdot\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}), \quad \boldsymbol{x}\in\mathbb{R}^N, \tag{1}$$

with mean $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{x}]$ and covariance matrix $\Sigma = \mathbb{E}[(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^T] > 0$. The distribution ρ is a particular case of Boltzmann distribution for harmonic energy functional E. The normalization constant Z in the case is known explicitly:

$$Z = \int\limits_{\mathbb{R}^N} e^{-E[\boldsymbol{x}]}d\boldsymbol{x}, \quad \ln Z = \frac{N}{2}\ln(2\pi) + \frac{1}{2}\ln\det\Sigma.$$

In ML the distribution ρ defines a model. The model can be trained by fitting parameters of the model given some observed samples $\{\boldsymbol{x}^m\}_{m=1}^M$ of the distribution ρ. We follow the maximum likelihood principle, whose goal is to find parameters for which observed samples have highest joint probability. Introduce likelihood function $\mathcal{L}$, which expresses probability to obtain the observed samples assuming model parameters are known and the samples are independent:

$$\ln\mathcal{L} = \frac{1}{M}\sum_{m=1}^{M}\ln\rho(\boldsymbol{x}^m|\Sigma,\boldsymbol{\mu}) = -\frac{1}{M}\sum_m E[\boldsymbol{x}^m] - \ln Z.$$

The first addendum is mean value of energy, it can be computed explicitly:

$$\begin{aligned}\frac{1}{M}\sum_m E[\boldsymbol{x}^m] &= \frac{1}{M}\left[\frac{1}{2}\boldsymbol{x}^m\cdot\Sigma^{-1}\boldsymbol{x}^m - \boldsymbol{x}^m\cdot\Sigma^{-1}\boldsymbol{\mu} + \frac{1}{2}\boldsymbol{\mu}\cdot\Sigma^{-1}\boldsymbol{\mu}\right]\\ &= \frac{1}{2}\operatorname{Tr}(\Sigma^{-1}V) - \boldsymbol{a}\cdot\Sigma^{-1}\boldsymbol{\mu} + \frac{1}{2}\boldsymbol{\mu}\cdot\Sigma^{-1}\boldsymbol{\mu},\end{aligned}$$

where

$$\boldsymbol{a} = \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{x}^m, \quad V = \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{x}^m(\boldsymbol{x}^m)^T.$$

and we have used $\operatorname{Tr}(ABC) = \operatorname{Tr}(BCA)$.

According to maximum likelihood principle, the parameters Σ and $\boldsymbol{\mu}$ can be estimated by maximization of $\mathcal{L}$ given observed values $\{\boldsymbol{x}^m\}$. Since Σ and $\boldsymbol{\mu}$ are not subjected to constraints except of $\Sigma^T = \Sigma$, the minimum should be a stationary point. Derivative with respect to $\boldsymbol{\mu}$ can be readily obtained:

$$\frac{\partial\ln\mathcal{L}}{\partial\boldsymbol{\mu}} = -\Sigma^{-1}(\boldsymbol{\mu}-\boldsymbol{a}) = 0,$$

hence maximum likelihood estimate $\boldsymbol{\mu} = \boldsymbol{a}$. Derivative of the partition function Z with respect to the covariance matrix Σ coincide with Σ up to a constant:

$$\frac{\partial\ln Z}{\partial\Sigma^{-1}} = \frac{1}{2}\frac{\partial}{\partial\Sigma^{-1}}\ln\det\Sigma^{-1} = \frac{1}{2}\Sigma.$$

Solution of the stationarity condition with respect to the covariance matrix

$$\frac{\partial\ln\mathcal{L}}{\partial\Sigma^{-1}} = -\frac{1}{2}V + \boldsymbol{\mu}\boldsymbol{a}^T - \frac{1}{2}\boldsymbol{\mu}\boldsymbol{\mu}^T + \frac{1}{2}\Sigma = 0,$$

gives us the estimate of the remaining parameter:

$$\Sigma = V - \boldsymbol{\mu}\boldsymbol{\mu}^T.$$

The same estimate can be obtained from minimization of Kullback–Leibler divergence between distribution ρ and empirical distribution $\tilde{\rho}$ of samples $\boldsymbol{x}^m$:

$$D_{KL}(\tilde{\rho}|\rho) = H(\tilde{\rho},\rho) - H(\tilde{\rho}) = \frac{1}{2}\left\{\operatorname{Tr}(\tilde{\Sigma}^{-1}\Sigma) + (\tilde{\boldsymbol{\mu}}-\boldsymbol{\mu})\cdot\tilde{\Sigma}^{-1}(\tilde{\boldsymbol{\mu}}-\boldsymbol{\mu}) - N + \ln\frac{\det\tilde{\Sigma}}{\det\Sigma}\right\},$$

where $\tilde{\mu}$ and $\tilde{\Sigma}$ are parameters of distribution $\tilde{\rho}$. Entropy is given by

$$H(\tilde{\rho}) = \mathbb{E}_{\tilde{\rho}}[-\ln\tilde{\rho}] = \frac{N}{2} + \frac{N}{2}\ln(2\pi) + \frac{1}{2}\ln\det\tilde{\Sigma} = \frac{N}{2} + \ln\tilde{Z}.$$

Cross-entropy can be estimated by Monte-Carlo method:

$$H(\tilde{\rho}, \rho) = \mathbb{E}_{\tilde{\rho}}[-\ln \rho] \approx -\frac{1}{M} \sum_{m=1}^{M} \ln \rho(\boldsymbol{x}^m) = -\ln \mathcal{L}.$$

Hence Kullback–Leibler divergence is equal to minus log-likelihood up to a constant independent of model parameters, which proves equivalence of maximum likelihood estimate and minimization of KL-divergence:

$$D_{KL}(\tilde{\rho}|\rho) = -\ln \mathcal{L} + \text{const}.$$

The training problem (calculation of Σ and $\boldsymbol{\mu}$ given $\boldsymbol{x}^m$) can be solved analytically, since the optimization problem is reduced to the solution of the linear system. For more complex ML problems it is convenient to use iterative methods, which increases likelihood function step by step until a local maximum is reached. The simplest iterative method is gradient ascend method with training step η:

$$\Sigma \mapsto \Sigma + \eta \frac{\partial \mathcal{L}}{\partial \Sigma}, \quad \boldsymbol{\mu} \mapsto \boldsymbol{\mu} + \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}}.$$

This update rule defines a pseudo-dynamics of the system. Here we are going to demonstrate that the dynamics can be turned to relaxation of the system with respect to control parameters, if we slightly modify the problem to take into account multiple samples in the training set. We consider two approaches: spatial sampling and temporal sampling. Doing spatial sampling we create M copies (equals to number of samples) of the initial system with the energy E such that all copies share the same parameters Σ, $\boldsymbol{\mu}$. Then energy of the training system is given by:

$$\tilde{E} = \frac{1}{M} \sum_{m=1}^{M} E[\boldsymbol{x}^m, \boldsymbol{\mu}, \Sigma].$$

We used normalization multiplier $1/M$ to closely match results of the maximum likelihood principle, but any multiplier leads to the same result. Derivative of the energy:

$$\frac{\partial \tilde{E}}{\partial \boldsymbol{\mu}} = \frac{1}{M} \sum_{m=1}^{M} \Sigma^{-1}(\boldsymbol{\mu} - \boldsymbol{x}^m) = \Sigma^{-1}(\boldsymbol{\mu} - \boldsymbol{a}) = -\frac{\partial \ln \mathcal{L}}{\partial \boldsymbol{\mu}},$$

hence indeed maximization of the likelihood with respect to the biases $\boldsymbol{\mu}$ is equivalent to minimization of energy with respect to the same parameters. This observation opens new possibilities for devices implementing distribution learning in hardware by dissipation of energy. For that purpose the energy should contain exchange $\boldsymbol{\mu} \cdot \Sigma^{-1} \boldsymbol{\mu}$ between degrees of freedom corresponding to parameters $\boldsymbol{\mu}$ and observable variables $\boldsymbol{x}$. For any system with continuous dependence of the energy on the state, the energy can be approximated by harmonic potential near metastable states. Hence until perturbations created by training samples stay small, the system can be trained as described above. Moreover, the exchange term of the required form appears in the Ising and the Heisenberg models of spin systems, in the models the perturbations do not have to be small. The BM based on the Ising model will be considered in Section 3, and BM based on the Heisenberg model is studied in Section 5.

The trained distribution sampling can be done by measuring states of the system while it is in thermal equilibrium. Time interval between observations should be large enough to make measurements uncorrelated. The temperature of the system controls variance. It is common in ML to simulate the system numerically by doing Gibbs sampling. Implementation of the system as a physical device can speed up sampling. If parameters $\boldsymbol{\mu}$, Σ are part of the design of the device, then to change distribution a new device should be designed and fabricated, which can be expensive and is not practical. However, if $\boldsymbol{\mu}$ are new degrees of freedom of the device, and the degrees of freedom can be controlled by external impulses, then the device is of general purpose and is reusable.

The strength of the physical implementation of the device is most clearly manifested in automatic fitting of the parameters to match an example distribution. The parameters of the system can be fit without usage of any external devices, only training examples should be provided. Sticking to spatial approach to training, let consider new system, which contains multiple copies of the system used for prediction, and let assume that the subsystems do not interact, except that they share degrees of freedom storing parameters $\boldsymbol{\mu}$. The training examples are provided by fixing degrees of freedom corresponding to output value $\boldsymbol{x}$, so that the state of $\boldsymbol{x}$ equals to $\boldsymbol{x}^m$ on copy m of the system. As was shown above, relaxation of the system with respect to unfixed parameters $\boldsymbol{\mu}$ is equivalent to maximization of likelihood of sampling the training set from the distribution generated by the system. The parameter $\boldsymbol{\mu}$ can be measured after relaxation, and the consumer device can have $\boldsymbol{\mu}$ be fixed by design, if the device is used only for prediction. On the other hand, the system can be used for online training, then training data is updated according to the current situation, and the prediction

is done by some copies of the system, which still share the parameters with other copies, but whose degrees of freedom $\boldsymbol{x}$ are not fixed and are subjected to stochastic dynamics.

Creation of multiple copies of the device can be a tricky engineering task, which also leads to increased size of the device. The temporal approach to training allows us to avoid these problems. Suppose we have a single exemplar of the system. At the moment t_m of time we apply an external control to fix state $\boldsymbol{x}^m$ of the system, then after a fixed interval we switch to the next sample $\boldsymbol{x}^{m+1}$ at the time t_{m+1}. The time derivative of state on each step is given by $\dot{\boldsymbol{\mu}}_m = \partial E[\boldsymbol{x}^m, \boldsymbol{\mu}]/\partial \boldsymbol{\mu}$, the mean velocity is obtained by averaging the derivatives over M steps

$$\frac{1}{M}\sum_{m=1}^{M}\frac{\partial E[\boldsymbol{x}^m, \boldsymbol{\mu}]}{\partial \boldsymbol{\mu}}.$$

which coincides with the velocity in the spatial approach. In contrast to the spatial approach, at each moment the dynamics is determined by one term $\dot{\boldsymbol{\mu}}^m$. However, if time steps $t_{m+1} - t_m$ are infinitesimal, the two dynamics coincide. In practice relaxation time over $\boldsymbol{\mu}$ should be much larger than the time step. The training either can be made once and the results of the training can be measured and used in another device dedicated only for prediction, or the training can be a continuous process with online update of parameters. In the second case, prediction steps are inserted among the training steps, such that the state $\boldsymbol{x}$ is allowed to evolve freely. To avoid drift of the parameters during the prediction phase, the relaxation time with respect to $\boldsymbol{\mu}$ should be significantly larger than relaxation time of $\boldsymbol{x}$.

3. Boltzmann machine

BM is an imaginary device, which state is random, and its distribution can be tuned by the training process. The state of BM is described by discrete random variables x_j, in the classical form only two values $\{0, 1\}$ are allowed for the variables. Except for the discrete nature of the state $\boldsymbol{x} = (x_j)_{j=1}^N$, all other parts of BM are very similar to the model described in Section 2. The constraints on values of $\boldsymbol{x}$ however make the problem harder to analyze and explicit solutions are not known except for very simple connection matrices. The restrictions also create multiple minima, making the model more expressive.

Main characteristic of BM is its energy, which coincides with the energy of the Ising model. It is customary to write energy of BM in the following form

$$E = \frac{1}{2}\boldsymbol{x}\cdot A\boldsymbol{x} - \boldsymbol{x}\cdot\boldsymbol{\mu},$$

which coincides with (1) up to an abandoned term quadratic with respect to $\boldsymbol{\mu}$. Since the variables x_j are discrete, the energy is bounded for all values of parameters A and $\boldsymbol{\mu}$. The joint distribution of the variables is the Boltzmann distribution:

$$p(\boldsymbol{x}) = \frac{1}{Z}e^{-E[\boldsymbol{x}]}, \quad Z = \sum_{\boldsymbol{x}} e^{-E[\boldsymbol{x}]}.$$

We drop from notation the parameters A and $\boldsymbol{\mu}$, which are assumed constant here. The abandoned term $\boldsymbol{\mu}\cdot A\boldsymbol{\mu}/2$ in the energy does affect normalization constant Z, but otherwise does not change behavior of BM in any way unless we do consider dynamics with respect to the parameters A and $\boldsymbol{\mu}$.

BM can express complex relationship between variables, if it is augmented with hidden units. Then the state vector is divided into two parts $\boldsymbol{x} = (\boldsymbol{v}, \boldsymbol{h})$. All the variables $\boldsymbol{x} = (\boldsymbol{v}, \boldsymbol{h})$ are separated to two classes: $\boldsymbol{v}$ are visible variables and $\boldsymbol{h}$ are hidden ones. The model should recreate probability distribution of only visible variables $\boldsymbol{v}$

$$p(\boldsymbol{v}) = \sum_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}), \quad p(\boldsymbol{v}, \boldsymbol{h}) = p(\boldsymbol{x}),$$

hence the training data contains only values of visible variables $\boldsymbol{v}^m$. Training of BM is done by maximization of the likelihood $\mathcal{L}$, which is probability of observation of samples $\boldsymbol{x}^m$ from the training set for given parameters A and $\boldsymbol{\mu}$:

$$\ln\mathcal{L} = \frac{1}{M}\sum_{m=1}^{M}\ln p(\boldsymbol{v}^m|A, \boldsymbol{\mu}) \to \max. \tag{2}$$

In practice BM is simulated by Gibbs sampling, and the likelihood is statistically estimated. For unconstrained BM the estimate is sufficiently good only for very few variables, resulting in a low training rate. The training can be significantly accelerated, if connections between variables in A are restricted in such a way that visible variables can not interact with themselves, and the same is true for the hidden variables. BM satisfying the condition is called restricted Boltzmann machine (RBM). The energy of RBM has the following form,

$$E = -\boldsymbol{v} \cdot W\boldsymbol{h} - \boldsymbol{c} \cdot \boldsymbol{h} - \boldsymbol{d} \cdot \boldsymbol{v}.$$

where W is the interaction matrix, $\boldsymbol{c}$ and $\boldsymbol{d}$ are biases for hidden and visible units respectively. The maximum of the likelihood can be computed by gradient ascend method with some small learning rate $\eta > 0$:

$$a \mapsto a + \eta \frac{\partial \ln \mathcal{L}}{\partial a},$$

applied for all parameters $a = \boldsymbol{c}, \boldsymbol{d}, W$. The derivatives can be computed explicitly:

$$\frac{\partial \ln \mathcal{L}}{\partial a} = G_a^+ - G_a^-, \quad G_a^+ = \frac{1}{M} \sum_{m=1}^{M} \frac{\partial}{\partial a} \ln \left(\sum_{\boldsymbol{h}} e^{-E[\boldsymbol{v}^m, \boldsymbol{h}]} \right), \quad G_a^- = \frac{\partial}{\partial a} \ln \left(\sum_{\boldsymbol{v}, \boldsymbol{h}} e^{-E[\boldsymbol{v}, \boldsymbol{h}]} \right).$$

The positive gradient G_a^+ can be expressed as average over all samples from the training set of the expectation value of the $-\partial E/\partial a$ over the hidden units:

$$G_a^+ = -\frac{1}{M} \sum_{m=1}^{M} \sum_{\boldsymbol{h}} p(\boldsymbol{v}^m|\boldsymbol{h}) \frac{\partial E[\boldsymbol{v}^m, \boldsymbol{h}]}{\partial a} = \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}\left[-\frac{\partial E}{\partial a} \middle| \boldsymbol{v}^m \right].$$

The negative gradient G_a^- is similar to G_a^+, but the expectation value is taken with respect the the model distribution of $\boldsymbol{v}$, instead of average over samples from the training set:

$$G_a^- = \mathbb{E}[-\partial E/\partial a].$$

The expectation values can be computed as sum over all outcomes for small BM. For a large number of variables the expectation values are estimated by Monte-Carlo method.

BM has a wide variety of uses, most prominent as associative memory and generative models. When used as an associative memory, BM is provided with partial (or noisy) input, that is only part of visible variables are set to the correct values in the initial state. Running simulation of BM the energy of the state is relaxed and BM is found in one of the remembered states, which is close to the initial state. The final state is a restored state associated with the initial state.

The visible variables can contain arbitrary information, for example we can split the variables into two parts: the first contains features, and the second contains labels. Given features, BM can assign labels by association. Thus BM can be used to solve classification problems.

4. Classification problem

The classification is a problem of assigning labels l to objects, described by their features $\boldsymbol{\phi}$. To take into account disputable cases, it is convenient to compute probability distribution over labels, instead of labels themselves. The probability distribution can be differentiated in contrast to discrete labels, which is necessary for many ML algorithms. Every mapping from the features f to the probability distributions over the labels l is called a classifier. We will denote $f(\boldsymbol{\phi})_l$ probability of $\boldsymbol{\phi}$ to belong to the class l. Doing ML we assume that the classifier belongs to a class of functions parameterized by vectors $\boldsymbol{\theta}$, $f = f(\boldsymbol{\phi}; \boldsymbol{\theta})$. The parameters $\boldsymbol{\theta}$ can be found by minimizing a loss function L, used to compare the model prediction and values from the training set $(\boldsymbol{\phi}^m, l^m)$. The predicted probability distribution $p = f(\boldsymbol{\phi}; \boldsymbol{\theta})$ for fixed $\boldsymbol{\phi}$ can be compared with correct distribution $\tilde{p}$ given by its samples $\{\tilde{l}^k\}_{k=1}^K = \{l^m \colon \boldsymbol{\phi}^m = \boldsymbol{\phi}\}$ using the following estimate for the cross-entropy:

$$H(\tilde{p}, p) = \mathbb{E}_{\tilde{p}}[-\ln p] \approx -\frac{1}{K} \sum_{k=1}^{K} \ln p_{l^k}.$$

Averaging the estimate over all inputs $\boldsymbol{\phi}$ we obtain common definition for loss function:

$$L_{KL}[\theta] = -\frac{1}{M} \sum_{m=1}^{M} \ln f(\boldsymbol{\phi}^m; \boldsymbol{\theta})_{l^m}, \tag{3}$$

which coincides with Kullback–Leibler (KL) divergence between model prediction and the distribution of training samples up to a constant. For comparison we also used mean square error (MSE) estimate:

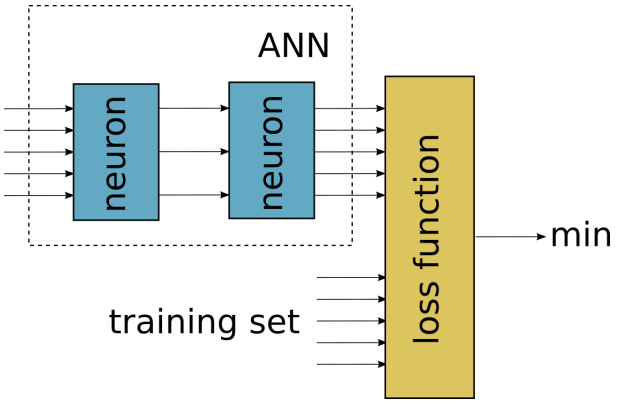

FIG. 1. Schematic of ANN composed of several layers, each representing a simple nonlinear transform with some parameters. Parameters are fitted by minimization of a loss function, which compares prediction of ANN with the training set containing ground truth data

$$L_{MSE}[\theta] = \frac{1}{MZ}\sum_{m=1}^{M}\sum_{k=1}^{Z}|f(\boldsymbol{\phi}^m;\boldsymbol{\theta})_k - \delta_{l^m,k}|^2,$$

where Z is the number of different labels. The optimal parameters $\boldsymbol{\theta}$ can be found by minimization of the loss:

$$\mathrm{argmin}_{\boldsymbol{\theta}} L[\boldsymbol{\theta}],$$

which can be done e.g. by gradient descent with the training step size constant η:

$$\boldsymbol{\theta} \mapsto \boldsymbol{\theta} - \eta\frac{\partial L}{\partial \boldsymbol{\theta}}.$$

4.1. Artificial neural network

One of the most popular models in ML is ANN, see Fig. 1 in the case, where $f = f_1 \circ \dots f_D$ is a composition of functions, called layers, of the form

$$f_d(\boldsymbol{\phi}) = a_d(\boldsymbol{\psi}), \quad \psi_k = \sum_j A_{d,kj}\phi_j + B_{d,k},$$

where A_d is a matrix, B_d is a bias, and a_d is an activation function. Since our ANN should return probability distribution, the activation function of the last layer is the softmax function

$$a_D(\boldsymbol{\psi})_k = \mathrm{softmax}(\boldsymbol{\phi})_k = \frac{e^{\psi_k}}{\sum_k e^{\psi_j}}.$$

Common choice for other layers is sigmoid functions, e.g.

$$a_d(\boldsymbol{\psi})_k = \sigma(\boldsymbol{\psi})_k = \frac{1}{2} + \frac{1}{\pi}\arctan\psi_k.$$

The parameters $\boldsymbol{\theta}$ for ANN consists of matrices and biases of linear layers:

$$\boldsymbol{\theta} = (A_1, B_1, \dots, A_D, B_D).$$

The gradient of the loss function with respect to parameters, required for training, is computed using backward propagation:

$$\frac{\partial L}{\partial A_d} = \frac{\partial L}{\partial \boldsymbol{y}}\frac{\partial f_D}{\partial \boldsymbol{x}_{D-1}}\cdots\frac{\partial f_{d+1}}{\partial \boldsymbol{x}_{d+1}}\frac{\partial f_d}{\partial A_d},$$

where $\boldsymbol{y} = L(\boldsymbol{x}_D)$, $\boldsymbol{x}_{d+1} = f_d(\boldsymbol{x}_d)$, $\boldsymbol{x}_1 = \boldsymbol{\phi}$. The expression for B_d is analogous.

As the most basic benchmark for the classifier, consider binary classification problem that maps single value $\phi \in [-1, 1]$ to one of two classes $\{0, 1\}$. We assume that all values ϕ larger certain threshold ϕ_0 belongs to the class 1, and the

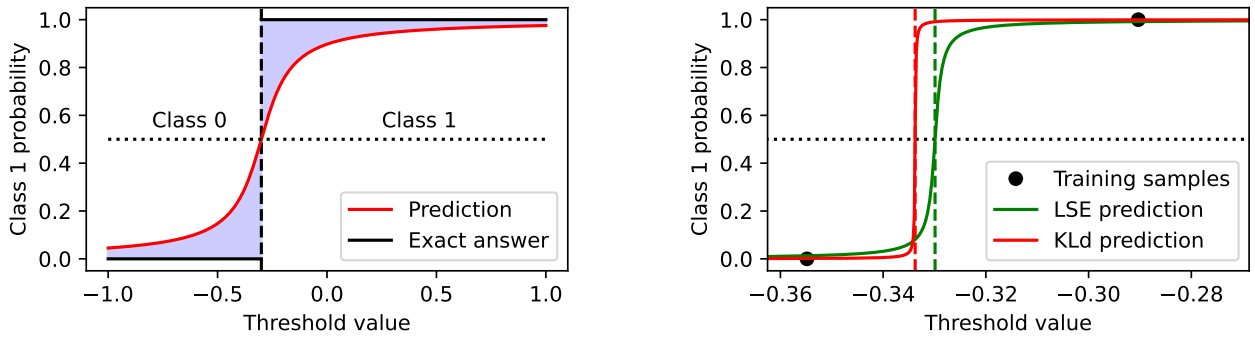

FIG. 2. (left) Binary classification problem with single input. All values smaller than the threshold belong to class 0, all other to class 1. The ML model predicts the probability of a value belonging to class 1. (right) Prediction made by ANN trained on 30 samples using MSE and KL-divergence loss functions. Only a region, where predictions are divergent from the exact answer, is shown

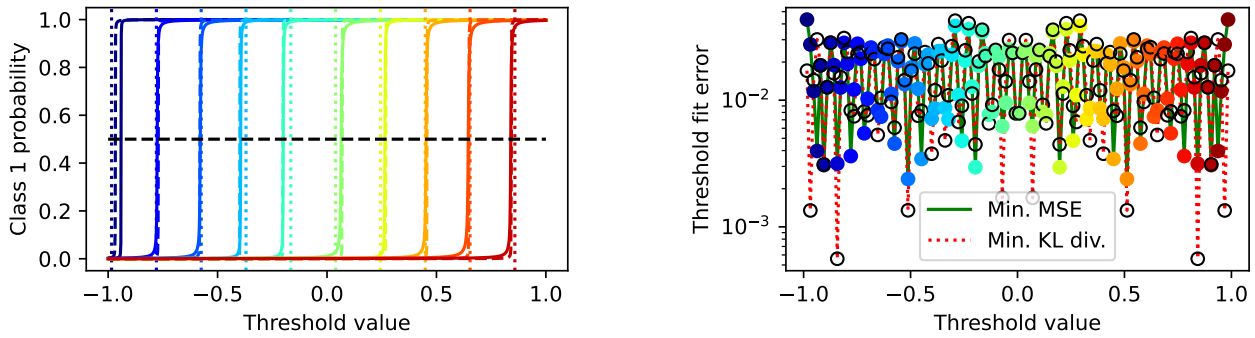

FIG. 3. (left) Predictions of ANN for the benchmark classification problem for different values of the threshold (shown by dotted vertical lines). Solid lines correspond to MSE loss, dashed line, to KL-divergence. (right) Error of fitted threshold values for MSE (filled color circles) and KL divergence losses

smaller values to the class 0. The probability distribution for the binary classifier is completely defined by the probability $p_1 = f(\phi;\theta)$ of the specimen belonging to the class 0. The ideal classifier in the case is defined by the function

$$f_0(\phi) = \begin{cases} 0, & \phi \le \phi_0, \\ 1, & \phi > \phi_0. \end{cases}$$

This step function, despite its simplicity, is not suitable for training, since it has zero gradient, except single point, where it is not differentiable. An appropriate model is an approximation of the step function with a sigmoid function σ, we use the following model:

$$f(\phi;\theta_1,\theta_0) = \sigma(\theta_1\phi + \theta_2).$$

Then the threshold value can be approximated by $\phi_0 \approx -\theta_2/\theta_1$. The left panel in Fig. 2 demonstrates both ideal classifier and the model one.

To demonstrate fitting of the threshold parameter, we train our model on a set of samples containing $M = 32$ uniformly distributed on the interval $[-1, 1]$ values ϕ^m, and the corresponding labels $l^m = f_0(\phi^m)$. The loss function is defined by Equation (3). The output of the trained model is shown on the right pane of Fig. 2. As can be seen, the model does not have enough data to find precise value of the threshold, instead the approximate threshold value is put on the maximum distance from point of both classes, which is not surprising, since our model is also a particular case of a support vector machine.

We repeated the training for different threshold values and estimated accuracy. The result is shown in Fig. 3. The metrics achieves 100% accuracy, except for the point $\phi_0 = 1$, where a single point belongs to the class 1. Both KL-divergence and MSE losses demonstrate very similar performance. The error of estimation of the threshold value ϕ_0 is within resolution of the training data for all values of ϕ_0.

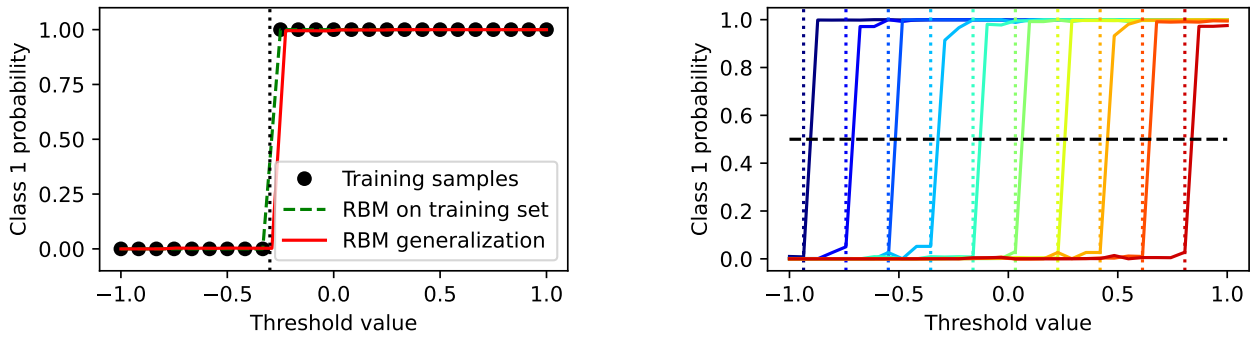

FIG. 4. (left) Prediction of RBM trained on 24 samples shown as black dots. Dotted vertical line represents exact threshold value. Prediction computed on the training set is represented by green dashed lines. Prediction on 32 samples not available in the training set is shown by the red solid line. (right) Prediction of RBM model for different values of threshold indicated by vertical dotted lines

Our trivial problem is solved exceptionally well by the very simple ANN, more complex networks are capable of solving problems of artificial intelligence. Both prediction and training of ANN are done in modern time on GPU or TPU devices, capable of computing arbitrary functions. The digital devices can compute the gradient with a high precision, but the high precision is not important for ANN training, therefore a recent trend is to reduce precision of computing to trade speed and reduce power consumption. Since ANN computation is analog in nature, it would be beneficial to implement it as an analog computer. However, it is hard to find a physical system that implements arbitrary activation functions and linear algebra naturally. Instead BM has natural physical counterparts and can be implemented directly in hardware more effectively than in simulation. The flexibility may be an issue. In the next section we will show, how to adapt BM easily to different problems.

4.2. Restricted Boltzmann machine

There are two common approaches to use RBM for classification. In the first approach RBM is used as a feature extractor, which maps visible variables to the latent variables, which are processed by a simple classifier, such as logistic regression. We will use the second approach, where the classification problem is solved solely by RBM. The visible variables $\boldsymbol{v} = (\boldsymbol{f}, l)$ are separated to the vector of features $\boldsymbol{f}$ and the label l (we restrict ourselves to binary classification problems, hence $l \in \{0, 1\}$). To encode the variable $x \in [-1, 1]$ as a bit string $\boldsymbol{f} \in \{0, 1\}^R$, we map x to the unit interval by $y = (x + 1)/2$, and let $\boldsymbol{f}$ be first R digits in the binary representation of y:

$$y = \sum_{l=1}^{L} 2^{-l} f_{l-1} + r, \quad |r| < 2^{-L}.$$

The distribution of the random pairs $(\boldsymbol{f}, l)$ generated by RBM can be fitted according to the training set (x^m, l^m). In the considered case the samples x^m are given by their approximated values $\boldsymbol{f}^m$ known with the resolution R. When we solve classification problem, we do not interested in the distribution of $\boldsymbol{f}$ itself, instead we are interested in the conditional distribution $p(l|\boldsymbol{f})$, hence the likelihood function (2) should be replaced by the conditional one:

$$\ln \mathcal{L} = \frac{1}{M} \sum_{m=1}^{M} \ln p(l^m | \boldsymbol{f}^m, A, \boldsymbol{\mu}) \to \max.$$

We made tests with RBM for the same binary classification problem as above with the resolution $L = 5$, which gives 32 different values of x. A handmade implementation of RBM in Python+NumPy was used. Total number of visible neurons was 6, namely $R = 5$ for features plus 1 for the label. We used three hidden units. All the training data were put into one batch. The RBM was trained 1000 iterations with step size 10, which resulted in the loss function value approximately 10^{-3}. The left pane of Fig. 4 demonstrates the result of the training on the dataset containing 25 uniformly distributed values of x^m out of 32 possible different values. The generalization result is perfect in the case and the threshold value is computed with precision of approximation of x by $\boldsymbol{f}$. We also tried to decrease the size of the training set, which however results in large errors at random values of $\boldsymbol{f}$. Since RBM initially has no information on the nature of the functions it is approximating, enough information should be provided to the machine to conclude that the function is a step function.

We repeated training of the RBM classifier for different threshold values ϕ_0 including all different values of $\boldsymbol{f}$ into the training set. The model predictions are shown in the right pane of Fig. 4, one line corresponds one value of ϕ_0, the

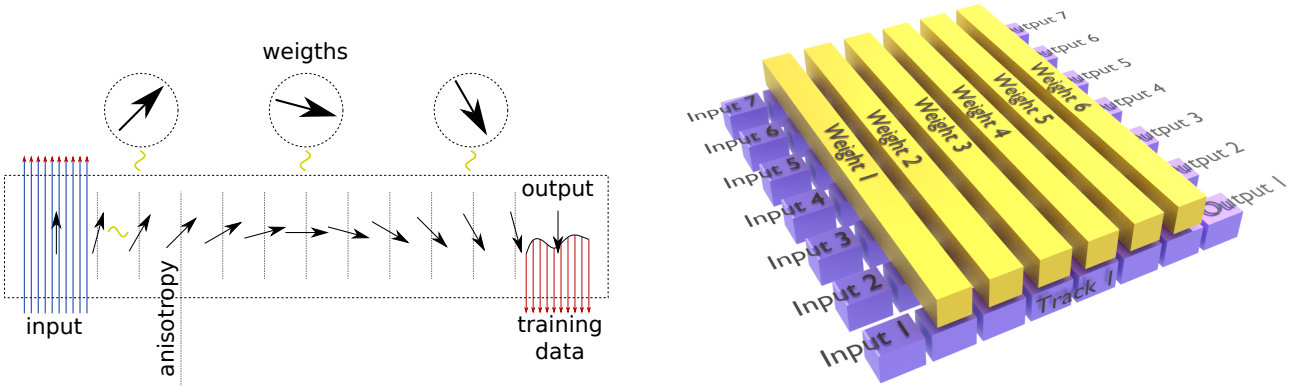

FIG. 5. (left) Schematic diagram of SBM consisting of chain of data spins, and weight spins associated with each data spin. Visible spins are attached to the ends and are controlled by the external magnetic field. Result is read from the rightmost spin. (right) Spatial approach to SBM training. Data spins are multiplied to match the number of training samples. Weight spins are shared by all copies of data spins

values themselves are shown as the vertical dotted lines. The classification was done without mistakes, both precision and recall were 100%.

The considered BM is an analog of the Ising model, hence each spin has one of two possible values $\{-1, 1\}$. The Heisenberg model allows the magnetic moment to point in arbitrary direction, hence is applicable for wide range of magnetic systems. In the next section we define a generalized BM based on the Heisenberg model.

5. Spin Boltzmann machine

Consider the Heisenberg model of a magnetic substance. The state of the system is defined by vector of magnetic moments $\boldsymbol{M}_j \in \mathbb{R}^3$, which are associated with atoms j or larger uniformly magnetized clusters. In the considered timescale the length M_j is constant, hence magnetization can be defined by its direction $\boldsymbol{M}_j = M_j \boldsymbol{S}_j$, $\|\boldsymbol{S}_j\| = 1$. The energy of the system includes the Heisenberg exchange energy:

$$E_{ex} = -\frac{1}{2} \sum_{j \neq k} J_{jk} \boldsymbol{S}_k \cdot \boldsymbol{S}_k,$$

where the exchange matrix J is symmetric, $J_{jk} = J_{kj}$. We also include easy axis anisotropy with axis $\hat{z}$ and constant $K_j > 0$, which makes states $\pm\hat{z}$ metastable for isolated spins, these states correspond to states ± 1 for BM or the Ising model:

$$E_{ani} = -\sum_j K_j S_{j,z}^2.$$

The last necessary contribution to the energy is Zeeman one, that is the energy of interaction with external magnetic field $\boldsymbol{B}$:

$$E_B = -\sum_j M_j \boldsymbol{B}_j \cdot \boldsymbol{S}_j.$$

We consider spins $\boldsymbol{S}_j$ a generalization of the variables x_j of BM. The energy $E = E_{ex} + E_{ani} + E_B$ is a close analog of energy of BM, and the energies coincide up to re-normalization if K_j and B_j are infinite. If a strong magnetic field is applied to a spin, the spin tends to rotate along the field, spiraling to the direction of the field due to the damping. Hence the external field $\boldsymbol{B}$ allows us to control individual spins or to define inputs of BM. For all other spins the field is set to zero that allows them to rotate freely.

The dynamics of the system is defined by stochastic Landau-Lifshitz-Gilbert equation [46]:

$$\dot{\boldsymbol{S}}_j = \frac{\gamma}{\mu_j} \boldsymbol{S}_j \times \left(-\frac{\partial E}{\partial \boldsymbol{S}_j} + \dot{W}_j \right) + \gamma\alpha \boldsymbol{S}_j \times \frac{d\boldsymbol{S}_j}{dt},$$

where γ is the gyromagnetic ratio, $\alpha > 0$ is the dissipation constant and $W_j = W_j(t)$ a Wiener process. Assuming the external field is constant, the probability density of states tends to the Boltzmann distribution:

$$\rho(S) = \frac{1}{Z} e^{-\beta E[S]}, \quad Z = \int_{\mathbb{S}^2} d\boldsymbol{S}_1 \cdots \int_{\mathbb{S}^2} d\boldsymbol{S}_N \cdot e^{-\beta E[S]},$$

where β is the inverse temperature. Although dynamics of the Heisenberg model is different from BM or Hopfield networks, the final distribution is the same, hence the Heisenberg model can be used in ML in the same way as BM. We call the BM based on the Heisenberg model spin Boltzmann machine (SBM).

We split all spins to three groups: visible I_v, hidden I_h and weights I_w; restriction of vector field S to spins from a group I_k we denote S^k, e.g. $S^v = \{\boldsymbol{S}_j : j \in I_v\}$. The first two groups are a direct analog of visible and hidden variables in BM, we will call them data spins $I_d = I_v \cup I_h$. The last group is an analog of parameters $\boldsymbol{\mu}$ in the model (1). We include some parameters of BM as degrees of freedom in our system, which allows us to train the model on new data and to reuse the device for new problems.

The visible spins are used as inputs and outputs, therefore the external field is applied only to these spins. If SBM functions in associative memory mode, the external magnetic field is only applied to the spins, which input values are known, all other spins rotate freely to relax the energy of the system. Magnetic field generated by the relaxed state is measured and decoded as an output of the SBM. For the classification problem we split the visible spin I_v to the group of inputs I_{in} and outputs I_{out}. The features are fed to SBM by applying an external field to spins I_{in}. In the prediction mode the external field is zero on the outputs I_{out}, and the probability distribution over labels are read from the spins I_{out}. In the training mode both spins I_{in} and I_{out} are constrained by the external field to match a sample from the training set.

Weights spins should interact with data spins, but the interaction can be quite arbitrary. We restrict ourselves to a simple case, where every data spin $j \in I_d$ is associated with exactly one weight spin $w(j) \in I_w$, and the weight spin interacts only with the corresponding data spin. The data spins interact with each other, the interaction matrix defines correlation between the variables. Taking into account specialization of all spins, the energy of the system can be expressed in the form $E = E_d[\boldsymbol{S}^d, \boldsymbol{S}^w] + E_w[\boldsymbol{S}^w]$, where energy of data spins is given by:

$$E_d[\boldsymbol{S}^d, \boldsymbol{S}^w] = -\sum_{j\in I_v} \mu_j \boldsymbol{B}_j \cdot \boldsymbol{S}_j - \sum_{j\in I_d} K_j S_{j,z}^2 - \sum_{j\in I_d} J_j^w \boldsymbol{S}_j \cdot \boldsymbol{S}_{w(j)} - \frac{1}{2} \sum_{\substack{j,k\in I_d \\ j\neq k}} J_{ik}^d \boldsymbol{S}_j \cdot \boldsymbol{S}_k.$$

The energy of self-interaction of weight spins E_w will be discussed below.

If dissipation constant is assumed to be large, then the relaxation dominates oscillations, and LLG equation can be approximately written in the following form:

$$\dot{\boldsymbol{S}}_j = \frac{\gamma\alpha}{\mu_j} \boldsymbol{S}_j \times \boldsymbol{S}_j \times \frac{\partial E}{\partial \boldsymbol{S}_j}. \tag{4}$$

The state of both data spins S^d and weight spins S^w are changed by the dynamics. However, we require magnetic moments M_j to be much larger for weights $j \in I_w$ than for data $j \in I_d$ to ensure that variation of weight is much slower than for data spins.

We distinguish two modes of operation of SBM: prediction and training modes. In the prediction mode SBM assigns a label to the object described by its features. In the mode the weight $\boldsymbol{S}^w$ are assumed to match the problem before running the algorithm, that is SBM should be trained in advance. The algorithm of usage of SBM in the prediction mode is as follows:

(1) Apply an external magnetic field to the input spins I_{in} according to the provided features.
(2) Let SBM relax.
(3) Read magnetic field on output spins I_{out}, statistics of the measurements defines probability distribution over labels.

Computation time must be much shorter than the relaxation time of the weight spins, but the second step must be longer than the relaxation time of the data spins, which can be achieved by appropriate choice of magnetic moments. If SBM is used only for prediction, the self-interaction energy of weights is not important, moreover the weight spins can be completely eliminated, and external magnetic field is substituted for the effective magnetic field produced by the weight spins. Without weights spins SBM coincides with BM generalized to the Heisenberg model.

Another mode of operation of SBM is the training mode. In the training mode we want to minimize average energy of SBM, where minimization is done with respect to the weight spins I_w, and the average is taken over samples from the training set fed to the visible spins I_v. The training can be done using spatial or temporal approaches (or mixture of both), introduced at the end of Section 2. In the spatial approach each sample of the training set is fed to its own copy of the data spins, herewith all copies share the same weight spins. Energy of the system for the spatial training

$$E = E_w[\boldsymbol{S}^w] + \frac{1}{M} \sum_{m=1}^{M} E_d[\boldsymbol{S}^{d,m}, \boldsymbol{S}^w].$$

The data samples are encoded by the external magnetic field on m-th copy of the visible spins $\boldsymbol{S}^{v,m}$. The external field is set constant, while the relaxation with respect to weight spins $\boldsymbol{S}^w$ is going on.

In the temporal approach a single exemplar of the system is enough, but data samples from the training set are fed one by one. In this approach the external magnetic field is not constant, but at each given moment encodes one sample from the dataset. The mean effective field acting on the weight spins is the same as in the spatial approach, hence relaxation dynamics are expected to be the same. The training algorithm for SBM in mixed spatial-temporal approach is as follows:

(1) Apply training samples to visible spins of all copies of the system. External magnetic field is applied to both inputs I_{in} and outputs I_{out} encoding features and labels respectively.
(2) Wait a period of time sufficient to relax data spins I_d.
(3) Repeat from step 1 with new training samples, until convergence of weight spins.

The algorithm resembles stochastic gradient descent commonly used for training on ANN. Recall that relaxation time for weight spins is much larger than for data spin, which is necessary for the convergence of the training iterations.

As was shown in section 2 the minimization of energy with respect to weights is equivalent to minimization of Kullback-Leibler divergence between predicted distribution and distribution of training data, at least if the state is defined by vectors from $\mathbb{R}^N$. On the other hand, the training can be considered as a process of memorization of the training data. Consider a simple case of one sample in the training set, and let there be no hidden spins. During the training phase, we find state of the weight spins $\boldsymbol{S}^w$ which minimize energy assuming fixed state of visible spins $\tilde{\boldsymbol{S}}^v$

$$\tilde{\boldsymbol{S}}^w = \operatorname{argmin}_{\boldsymbol{S}^w} E[\tilde{\boldsymbol{S}}^v, \boldsymbol{S}^w].$$

In the prediction phase, given weights obtained from the training, we expect that minimum of energy E with respect to the visible spins should revive training sample $\tilde{\boldsymbol{S}}^v$:

$$\tilde{\boldsymbol{S}}^v = \operatorname{argmin}_{\boldsymbol{S}^v} E[\boldsymbol{S}^v, \tilde{\boldsymbol{S}}^w]. \tag{5}$$

Thus for every vector $\tilde{\boldsymbol{S}}^v$ there should be a vector $\tilde{\boldsymbol{S}}^w$ such that E has minimum at the point $(\tilde{\boldsymbol{S}}^v, \tilde{\boldsymbol{S}}^w)$ with respect to both arguments. The simplest way to ensure the condition is to define energy as a symmetric function with respect to swap of arguments $\boldsymbol{S}^d$ and $\boldsymbol{S}^w$. Exchange term between data spins and weights is already symmetric. Since we have freedom of definition of self-interaction of weights E_w, we can define exchange constants to be equal to exchange constants for data spins:

$$E_w = -\sum_{\substack{j,k \in I_d \\ j \neq k}} J_{d,jk} \boldsymbol{S}_{w(j)} \cdot \boldsymbol{S}_{w(j)}.$$

Introduction of interaction between weights is not strictly speaking necessary to satisfy the condition (5). Since magnetic moments directions are subjected to normalization condition $\boldsymbol{S}_j^2 = 1$, the energy functional is always bounded, hence it has at least one minimum. During the training phase, the system is relaxed with respect to both data spins and weight spins, hence the result of the minimization should be minimum with respect to all degrees of freedom, which ensures the condition (5).

6. Benchmark

We check the performance of SBM on a simple binary classification problem introduced in Section 4. Internal structure of SBM in this case can be as simple as a chain of N spins, see the left panel in Fig. 5. Denote $\boldsymbol{S}_{0,j}$ the state of the data spin j, and $\boldsymbol{S}_{1,j}$ the state of the corresponding weight. For simplicity we assume interaction constant between data spins to be constant J. The easy axis anisotropy with axis $\hat{\boldsymbol{z}}$ and constant K is applied to the chain. Interaction between data spin and the corresponding weight is set to a constant value W. The weight spins do not interact with each other. All spins except the first and the last are hidden. The first spin is an input, and the last spin is an output. The control is applied as external magnetic field to the spin 1, and during the training phase to the spin N. We assume that magnetic moments of all data spins equal to a constant μ_0, and all magnetic moments of weight spins equal to a constant $\mu_1 \gg \mu_0$. The resulting energy functional is as follows:

$$E[S] = -J\sum_{j=2}^{N} \boldsymbol{S}_{0,j-1} \cdot \boldsymbol{S}_{0,j} - W\sum_{j=1}^{N} \boldsymbol{S}_{0,j} \cdot \boldsymbol{S}_{1,j} - K\sum_{j=1}^{N} S_{0,j,z}^2 - \mu_0(\boldsymbol{B}_0 \cdot \boldsymbol{S}_{0,0} + \xi \boldsymbol{B}_N \cdot \boldsymbol{S}_{0,N}).$$

Input value $\phi \in [-1, 1]$ is encoded in direction of the magnetic field as

$$\boldsymbol{B}_0 = B\phi\hat{\boldsymbol{z}} + B\sqrt{1-\phi^2}\hat{\boldsymbol{x}}.$$

The class l in the training data is mapped to the vector field as follows:

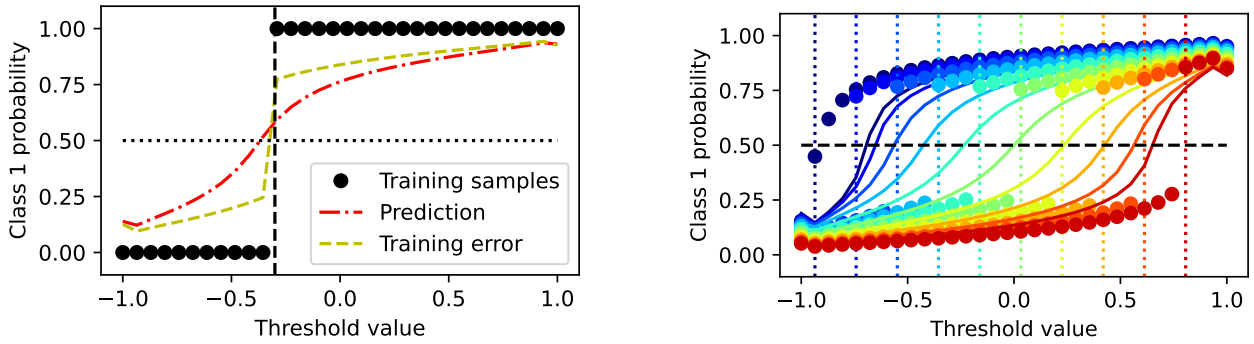

FIG. 6. (left) Output of SBM trained on 32 samples shown by black dots. The threshold value is marked by a vertical dashed line. Red solid line demonstrates model prediction. Yellow dash-dotted line shows output of SBM during the training. (right) SBM prediction (solid lines) for different values of the threshold (vertical dotted lines)

$$\boldsymbol{B}_N = \begin{cases} B\hat{\boldsymbol{z}}, & l = 1, \\ -B\hat{\boldsymbol{z}}, & l = 0. \end{cases}$$

The output of SBM is given by z projection of the last spin in the chain. The probability p_1 of class 1 is decoded in the following way, consistent with representation of the training data:

$$p_1 = 2S_{0,N,z} - 1. \tag{6}$$

We use the spatial approach for the training, creating M copies of the system (by number of samples in the training set), such that all copies share the same weights. One possible organization of the device is shown in the right pane of Fig. 5. The data spins are organized in parallel chains in bottom layer of the system, while weight spins lies in the upper layer. The weights are made of material having high magnetic stiffness to ensure their coherent rotation, each weight can be considered as a single spin having a large magnetic moment. The weights cross all chains containing data spins and interact with all of them with the same exchange constant. The chains of data spins do not interact with each other. We assume the small size of the system, then the demagnetizing field can be neglected. Since there is no special restriction on interaction between spins for general SBM, larger systems with dipole-dipole interaction should probably function as well as SBM, but this is a subject of further research.

We have selected the following parameters of SBM, which give deterministic result for random initial state of spins:

$$J = 3J_0, \quad W = 0.05J_0, \quad K = 0.07J_0, \quad \mu_0 B = J_0,$$

where J_0 is an arbitrary unit of energy. As a training set we use 32 uniformly distributed values ϕ^m on the interval $[-1, 1]$ and the corresponding label $l^m = \theta(\phi^m - \phi_0)$, where θ is Heaviside step function. The left panel in Fig. 6 demonstrates the result of the training for a fixed $\phi_0 = -0.3$. Each sample was fed to the corresponding copy of the spin chain by applying an external magnetic field both to the first and the last spins. Then the system was relaxed to minimize total energy over all degrees of freedom. The initial state of weight spins is random, hence the final state is also random. However, for the chosen values of parameters only one minimum is possible, hence the result is deterministic. We observed that if too strong magnetic field is applied, the orientation of inputs and outputs match exactly the direction of the field, but when the field is removed the end spin start rotating forgetting its target direction. Hence we used a moderate field $\xi = 0.08$. z-projection of the output spin is labeled as training output in Fig. 6. The output of SBM was computed once again on the same inputs doing prediction. The input ϕ^m was encoded to the first spin in every chain by applying the corresponding external field. No external field is applied to the last spin in the chains. The energy of the system is minimized assuming weights are fixed. The z-projection of the last spin in every chain is decoded as output, labeled in Figure as the prediction result. The output does not recover the training sample, but the threshold value is determined correctly.

To check performance of SBM for different threshold values, we repeated the training for 32 different values of $\phi_0 = \phi^m$. The result is shown in the right pane in Fig. 6. Each line corresponds to one threshold value, the vertical dashed line marks the corresponding ϕ_0. Dots show output of SBM in the training mode, that is orientation of the output spins decoded according to (6). Even when external field is applied to the output spin, its magnetic moment does not align with the magnetic field perfectly. The alignment can be enforced by a stronger training field, but after turning the field off the output spin will rotate to a larger angle, resulting in larger errors in prediction mode.

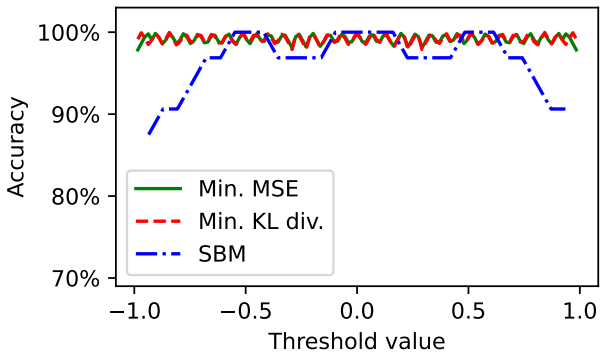

FIG. 7. Accuracy of classification by ANN with MSE loss (solid green line), ANN with KL-divergence loss (dashed red line) and SBM (dash-dotted blue line)

Metrics for the solution of the classification problem for ANN and SBM are shown in Fig. 7 on the left and right panels respectively. Accuracy of ANN is always greater 97%, while accuracy of SBM stays greater than 87%. The largest error occurs for extreme values of the threshold, where small variation of z-projection of the magnetic moment corresponds to large rotation of spin increasing overall error.

7. Conclusion

We extended BM by including weight spins, which play the role of biases. The Heisenberg model of a magnetic system is used, which is more accurate than the Ising model. We call the generalization SBM. SBM can be implemented as a device based on artificial spin ice. The weights can be tuned to enable SBM to solve different problems, therefore the device becomes reusable. We proposed a scheme and an algorithm of training of SBM on a given training set, which is in the essence relaxation with respect to weight spins, when inputs and outputs are fixed by the external magnetic field according to the training examples. If state space is a Euclidean space, then the minimization of energy is equivalent to minimization of Kullback-Leibler divergence. For magnetic systems, where magnetic moments are constrained, the exact optimization target is unknown, but BSM nevertheless can function as associative memory. We tested the performance of SBM on a simple binary classification problem and compared the results with the one of an ANN and standard BM. SBM was able to solve the problem, but the accuracy was lesser than the one of ANN and BM, since the exchange matrix in SBM was fixed, while all weights in ANN and BM were optimized. Due to the very simple design of SBM, small sizes and absence of power consumption for computations itself (power is consumed only for input-output), SBM can be an efficient realization of ML devices of the future.

Weight spins of SBM are a tunable analog of biases in BM, but exchange matrix is fixed both in SBM and in BM, that is exchange matrix should be precomputed by a training algorithm using general purpose computers and then fabricated during the manufacturing of the device implementing BM or SBM. The exchange matrix is crucial in determination of correlations between inputs and outputs. The relative poor performance of SBM in the benchmark can be addressed to the fact that the exchange matrix was not tuned. Exchanges between spins in real physical devices are harder to control than biases, however, we believe that this can be done using materials, which demonstrate three or four spins interaction [47]. If spins with large magnetic moments are added to such systems, in a short time scale their state can be considered as constant, but it will affect interaction between other spins.

In the studied benchmark we restricted ourselves to nearest-neighbors interactions. This assumption is valid only for nanoscale systems or near neighbor spin ice [48], where dipolar interactions do not manifest themselves. For larger devices all parts of the systems interact with each other due to the demagnetizing field. The general principles of operation of SBM do not restrict long range interactions between spins, hence the general approach should be applicable to dipolar spin ice, however their performance in this mode is a subject of another study.

In the proposed SBM, relaxation is used both for prediction and for training, but the training process takes orders of magnitude longer than prediction. Relaxation takes much longer than spin waves propagation through the system, hence significant speed up can be achieved, if the relaxation will be used only for training and prediction will be done by wave propagation. Logic devices based on spin-waves were previously demonstrated in [49]. These devices can be augmented with controlled scatterers, which can be fitted by ML methods.

References

[1] Alzubaidi L., Zhang J., Humaidi A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *J. Big Data*, 2021, **8**(53).
[2] Karniadakis G.E., Kevrekidis I.G., Lu L. et al. Physics-informed machine learning. *Nat Rev Phys*, 2021, **3**, P. 422–440.
[3] Hebb D.O. *The Organization of Behavior.* New York: Wiley & Sons.: 1949, 365 p.
[4] Marvin M., Papert S.A. Perceptrons: An Introduction to Computational Geometry. The MIT Press.: 1969, 292 p.

[5] Reuther A., Michaleas P., Jones M. AI and ML Accelerator Survey and Trends. 2022 IEEE High Performance Extreme Computing Conference (HPEC).
[6] Siegelmann H.T. *Neural Networks and Analog Computation: Beyond the Turing Limit*. Springer: 1999.
[7] Sun Z., Pedretti G., Ambrosi E. et. al. Solving matrix equations in one step with cross-point resistive arrays. *PNAS*, 2019, **116**(10), P. 4123.
[8] Hughes T.W., Williamson A.D., Minkov M., Fan Sh. Wave physics as an analog recurrent neural network. *Science Advances*, 2019, **5**(12).
[9] Grollier J., Querlioz D., Stiles M. D. Spintronic Nanodevices for Bioinspired Computing. *Proc IEEE Inst Electr Electron Eng.*, 2016, **104**(10), P. 2024–2039.
[10] Torrejon J., Riou M., Araujo F. et al. Neuromorphic computing with nanoscale spintronic oscillators. *Nature*, 2017, **547**, P. 428–431.
[11] Stephan A.W., Lou Q., Niemier M.T., Hu X.S., Koester S.J. Nonvolatile Spintronic Memory Cells for Neural Networks. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 2019, **5**(2), P. 67–73.
[12] Ostwal V., Zand R., De Mara R. A Novel Compound Synapse Using Probabilistic Spin–Orbit-Torque Switching for MTJ-Based Deep Neural Networks. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 2019, **5**(2), P. 182–187.
[13] Bläsing R., Khan A.A., Filippou P.Ch. Magnetic Racetrack Memory: From Physics to the Cusp of Applications Within a Decade *Proceedings of the IEEE*, 2020, **108**(8), P. 1303–1321.
[14] Sai L., Wang K., Xichao Zh. Magnetic skyrmions for unconventional computing. *Materials Horizons*, 2021, **8**(3), P. 854–868.
[15] Vakili H., Xu J-W., Zhou W. Skyrmionics—Computing and memory technologies based on topological excitations in magnets. *Journal of Applied Physics*, 2021, **130**, 070908.
[16] Song K. M., Jeong J-S., Pan B, et. al. Skyrmion-based artificial synapses for neuromorphic computing. *Nature Electronics*, 2020., **3**, P. 148–155.
[17] Chakravarty A., Mentinka J.H., Davies C.S., Yamada K.T., Kimel A.V., Rasinga Th. Supervised learning of an opto-magnetic neural network with ultrashort laser pulses. *Appl. Phys. Lett.*, 2019, **114**, 192407.
[18] Pinna D., Bourianoff G., Everschor-Sitte K. Reservoir Computing with Random Skyrmion Textures. *Phys. Rev. Applied*, 2020, **14**, 054020.
[19] Hopfield J.J. Neural networks and physical systems with emergent collective computational abilities. *PNAS*, 1982, **79**(8), P. 2554–2558.
[20] Hopfield J.J. Neurons With Graded Response Have Collective Computational Properties Like Those of Two-State Neurons. *Proceedings of the National Academy of Sciences*, 1984, **81**(10), P. 3088-3092.
[21] Hopfield J.J. Searching for memories, Sudoku, implicit check bits, and the iterative use of not-always-correct rapid neural computation *Neural Comput*, 2008, **20**(5), P. 1119-64.
[22] Sherrington D., Kirkpatrick S. Solvable Model of a Spin-Glass. *Physical Review Letters*, 1975, **35**(26), P. 1792.
[23] Hinton G., Sejnowsk T.J. Analyzing Cooperative Computation. 5th Annual Congress of the Cognitive Science Society. Rochester, New York, 1983.
[24] MacKay D. *Information Theory, Inference, and Learning Algorithms.* University of Cambridge, 2003.
[25] Neelakanta P., De Groff D.F. *Neural Network Modeling: Statistical Mechanics and Cybernetic Perspectives*. CRC Press, Boca Raton.: 1994. 256 p.
[26] Freeman W.J., Vitiello G. Dissipation and spontaneous symmetry breaking in brain dynamics. *J. Phys. A: Math. Theor*, 2008, **41**(30), P. 304042.
[27] Gori M., Maggini M., Rossi A. Neural network training as a dissipative process *Neural Networks*, 2016, **81**, P. 72–80.
[28] Möller M., Möllenhoff T., Cremers D. Controlling Neural Networks via Energy Dissipation. International Conference on Computer Vision (ICCV), 2019, P. 3255–3264.
[29] Fukami S., Ohno H. Perspective: Spintronic synapse for artificial neural network featured. *Journal of Applied Physics*, 2018, **124**, P. 151904.
[30] Saeedi M., Markov I.L. Synthesis and optimization of reversible circuits — a survey. *ACM Computing Surveys*, 2013, **45**(2), P. 1–34.
[31] Nielsen M.A., Chuang I.L. *Quantum Computation and Quantum Information.* Cambridge University Press. 2011.
[32] Ventra M.D., Traversa F.L. Perspective: Memcomputing: Leveraging memory and physics to compute efficiently. *Journal of Applied Physics*, 2018, **123**, P. 180901.
[33] Camsari K.Y., Faria R., Sutton B.M., Datta S. Stochastic p-Bits for Invertible Logic. *Phys. Rev. X*, 2017, **7**, P. 031014.
[34] Borders W.A., Pervaiz A.Z., Fukami S., Camsari K.Y., Ohno H., Datta S. Integer factorization using stochastic magnetic tunnel junctions. *Nature*, 2019, **573**, P. 390–393.
[35] Bearden S.R.B., Pei Y.R., Di Ventra M. Efficient solution of Boolean satisfiability problems with digital memcomputing. *Sci Rep*, 2020, **10**, P. 19741.
[36] Gypens P., Waeyenberge B.V., Di Ventra M., Leliaert J., Pinna D. Nanomagnetic Self-Organizing Logic Gates. *Phys. Rev. Applied*, 2021, **16**(2), P. 024055.
[37] Balynskiy M., Chiang H., Gutierrez D., Kozhevnikov A., Filimonov Yu., Khitun A. Reversible magnetic logic gates based on spin wave interference *Journal of Applied Physics*, 2018, **123**, P. 144501.
[38] Chauwin M., Hu X., Garcia-Sanchez F., Betrabet N., Paler A., Moutafis C., Friedman J.S. Skyrmion Logic System for Large-Scale Reversible Computation *Phys. Rev. Applied*, 2019, **12**(6), P. 064053.
[39] Amit D.J., Gutfreund H., Sompolinsky H. Spin-glass models of neural networks. *Phys. Rev. A*, 1985, **32**, P. 1007.
[40] Bramwell S.T., Harris M.J. The history of spin ice. *Journal of Physics: Condensed Matter*, 2020, **32**(37), P. 374010.
[41] Farhan A., Derlet P., Kleibert A., Balan A., Chopdekar R. V., Wyss M., Anghinolfi L., Nolting F., Heyderman L.J. Exploring hyper-cubic energy landscapes in thermally active finite artificial spin-ice systems. *Nature Phys*, 2013, **9**, P. 375–382.
[42] Kaffash M.T., Lendinez S., Jungfleisch M.B. Nanomagnonics with artificial spin ice. *Physics Letters A*, 2021, **402**, P. 127364.
[43] Jensen J.H., Folven E., Tufte G. Computation in artificial spin ice. Proceedings of the ALIFE 2018: Tokyo, Japan. July 23–27, 2018. (P. 15–22).
[44] Gypens P., Leliaert J., Van Waeyenberge B. Balanced Magnetic Logic Gates in a Kagome Spin Ice. *Phys. Rev. Applied*, 2018, **9**, P. 034004.
[45] Arava H., Leo N., Schildknecht D., Cui J., Vijayakumar J., Derlet P. M., Kleibert A., Heyderman L.J. Engineering Relaxation Pathways in Building Blocks of Artificial Spin Ice for Computation. *Phys. Rev. Applied*, 2019, **11**, P. 054086.
[46] Banas L., Brzezniak Z., Neklyudov M., Prohl A. *Stochastic Ferromagnetism: Analysis and Numerics.* De Gruyter. 2014.
[47] Hoffmann M., Blügel S. Systematic derivation of realistic spin models for beyond-Heisenberg solids. *Phys. Rev. B*, 2020, **101**, P. 024418.
[48] Bramwell S.T., Gingras M.J.P. Spin Ice State in Frustrated Magnetic Pyrochlore Materials. *Science*, 2001, **294**(5546), P. 1495–1501.
[49] Klinglera S., Pirrob P., Brächer T., Leven B., Hillebrands B., Chumak A.V. Spin-wave logic devices based on isotropic forward volume magnetostatic waves. *Appl. Phys. Lett*, 2015, **106**, P. 212406.

Submitted 26 October 2022; revised 17 November 2022; accepted 30 November 2022

Information about the author:

Igor S. Lobanov – Faculty of Physics, ITMO University, Lomonosova Str. 9, Saint Petersburg, 191002 Russia;
ORCID 0000-0001-8789-3267; igor.lobanov@metalab.ifmo.ru, lobanov.igor@gmail.com