Dual stochasticity of the cortex as a most efficient and biologically plausible learning algorithm

Jun-nosuke Teramae
Graduate School of Informatics, Kyoto University, Kyoto, Japan
teramae@acs.i.kyoto-u.ac.jp

Abstract

Neurons and synapses in the cerebral cortex behave stochastically even during precise perception and reliable learning of animals. While studies have revealed advantages of these stochastic features, relationship and synergy of them remain elusive. Here, we show that these stochastic features are unified into a framework to provide an efficient and biologically-plausible learning algorithm that consistently explains various experimental findings of the brain, which includes statistics of cortical circuit and the most efficient power-low encoding of cortex. The algorithm can be regarded as an integration of major branches of existing algorithms of neural networks, back propagation learning and Boltzmann machine, while it overcomes known limitations of them. As an experimentally testable prediction, slow retrograde modulation of excitability of neurons is also derived. We anticipate that the algorithm is, due to its simplicity and flexibility, beneficial to develop neuromorphic devices or scalable AI-chips, and bridges the gap between neuroscience and machine learning.

Introduction

In the cortex, neurons continuously generate irregular spike trains with fluctuating membrane potential and largely varying firing rate even across trials where an animal solves the same task and shows identical responses [1-7]. Besides neurons, synapses are also stochastic in cerebral cortical [8]. Formation, elimination and volume change of dendritic spines exhibit random fluctuation [9-15] and transmitter release from synapses is also inherently unreliable [16-18]. These stochastic features may realize nearly optimal computation and learning under uncertain environment via Bayesian inference [19-28].

So far, however, the stochastic behaviors of neurons and synapses are separately studied. It remains unclear why the both should be stochastic and what kind of neural computation

emerges from the synergy of them. It is also uncertain how behavioral response and learning can be precise and reliable under stochastic nature of neurons and synapses and why timescales of them are so different, millisecond order of spike firing of neurons and second or more than hours for synapses.

Here, we show that the stochastic behaviors of neurons and synapses are inseparably integrated into a simple framework of a sampling-based Bayesian inference, in which a synergy of them provides an effective and flexible learning algorithm that is consistent with various experimental findings of neurons and synapses in cortex. The derived algorithm well explains plasticity of cortical synapses [29, 30], reproduces extremely different timescales of neurons and synapses, higher-order statistics of topology of local cortical circuit [31], and response properties of cortical neurons including the nearly optimal power-low scaling of population activity [9]. These results seem to strongly suggest that the stochastic behaviors of neurons and synapses are inseparable substance of neural computation and learning. As an experimentally testable prediction, slow retrograde modulation of firing activity of neurons by postsynaptic neurons, which has not been explored yet, is also provided.

The derived algorithm seems as a natural integration of representative branches of learning theories, error backpropagation learning [32, 33], Bayesian inference [21], Boltzmann machine [34], and reinforcement learning [35, 36]. The algorithm provides a stochastic variant of the error backpropagation, and simultaneously is regarded as an extension of the Boltzmann machine. Unlike the backpropagation learning, however, it requires neither objective functions, fine-tuning of parameters, coordinated nor synchronous updates of variables, a feed-forward network structure, nor alternative execution of forward and backward computations [32]. Rather, repetition of locally and asynchronously performed stochastic update of neurons and synapses in a network realizes supervised and unsupervised learning. As the algorithm is not derived as an optimization of objective functions, it rarely shows serious overfitting. We also discuss close relationship between the algorithm and temporal difference (TD) learning [35] of reinforcement learning.

Results

Neural networks

Most of connections between neurons in cortex are redundantly expressed by multiple synapses [27, 37]. Introducing this redundancy and stochasticity of neurons and synapses, we model a neural network as a population of neurons connected with each other via multiple synapses, and describe each neuron and synapse in the network equivalently as a binary stochastic variable (Fig. 1a, Methods). A neural variable describes whether the neuron generates a spike whereas synaptic variable represents whether the synapse has its contact to a dendrite of the postsynaptic neuron. Connection weight from the ith to the jth neuron, that is generally asymmetric, is then given as the weighted sum of the synaptic states as $w_{ij} = \sum_{k=1}^{K} s_{ijk} a_{ijk}$, where $s_{ijk} = \{0,1\}$ is the state of the kth synapse of the connection from the ith to the jth neuron. A constant a_{ijk} represents contribution of the synapse to the weight, which may correspond to amplitude of miniature postsynaptic potential (PSP) of the synaptic contact. Note that a_{ijk} is a constant and does not change during learning. We model a neuron in the network, for now, as it receives inputs from presynaptic neurons and generate a spike with the probability $P(x_i = 1) = \sigma(v_i) = \sigma\left(\sum_j x_j w_{ij}\right)$, where x_i is the state of the ith neuron and $\sigma(x)$ is the activation function of a neuron, which we use the sigmoidal function $\sigma(x) = (1 + e^{-x})^{-x}$ through the paper. The weighted sum of the inputs v_i may correspond to membrane potential of the neuron.

External inputs, including target outputs of supervised learning, are presented to the network as variables of a part of neurons, visible neurons, are fixed to values of the data. The data, thus, should be represented as binary vectors. Other neurons in the network that do not receive external data directly are denoted as hidden neurons.

Biologically-plausible Learning

Under the above setting, we formulate learning of the network as a continuing Gibbs sampling [38] of all free variables, that includes all synapses and hidden neurons, in the network from the posterior distribution conditioned on external environment or given dataset $P\left(\{x_{di}\}_{i\in H, d\in D}, \{s_{ijk}\}_{i,j\in N, k\in K} \middle| \{x_{di}\}_{i\in V}\right)$ (Methods). In other words, we hypothesize that stochastic activities of neurons and synapses in the cortex is a continuing random process to generate samples of them that suitably explain external world. Here, d

is the data index of the given dataset D, V and H denote sets of visible and hidden neurons, N represents a set of all neurons, and K denotes a set of synapses for each connection. Note that states of neurons, including those of hidden neurons, depend on each data of the dataset, while those of synapses are shared by all data of the dataset and do not have data index because the aim of the learning is generally to obtain networks, i.e. sets of synaptic states, that consistently explain all data of the dataset (Fig. 1b, 1c).

The Gibbs sampling ensures that we can replace sampling from the posterior distribution that is computationally and biologically intractable due to high dimensionality and complex dependencies among variables with iterative samplings of each one variable from a posterior distribution of the variable conditioned on all other variables. Furthermore, due to flexibility of the Gibbs sampling, each sampling can be performed in arbitral order and frequencies, which means that each neuron and synapse can asynchronously update its variable with its own arbitral frequency without any global schedule or coordination among them.

Applying Bayes' theorem to posterior distributions and assuming that the amplitudes of synapses are small, we obtain stochastic dynamics, i.e. update rules, of neurons and synapses (Methods) as

$$P(x_{di} = 1 | \cdots) = \sigma(v_{di} + b_{di}) \tag{1}$$

$$b_{di} = \sum_j w_{ij} \left(x_{dj} - \sigma\left(v_{dj}\right) \right) \tag{2}$$

for a neuron and

$$P\left(s_{ijk} = 1 \middle| \cdots\right) = \sigma\left(q_{0,ijk} + a_{ijk} q_{ij}\right) \tag{3}$$

$$q_{ij} = \sum_d x_{di} \left(x_{dj} - \sigma\left(v_{dj}\right) \right) \tag{4}$$

for a synapse, where "⋯" is the abbreviation of "all variables other than the target variable" and $q_{0,ijk}$ is a constant representing a prior bias of the synaptic state, which we set $q_{0,ijk} = 0$ through the paper.

The above equations are the learning algorithm of the neural network. Following the equations, the state of each neuron and synapse asynchronously updates its state from time to time. As we will show later, this simple repetition of stochastic and asynchronous

updates of neuron and synapses allows the network to reproduce given dataset with high probability. For instance, if we apply the algorithm to a supervised learning in which input and output neurons are clumped to given input and desired output respectively, the network becomes to generate desired output with high probability even when only input data, that may not be included in the training dataset, is given to the input neurons.

Unlike the backpropagation learning in which alternative execution of the forward and the backward computations are required, the algorithm achieves learning by simple iteration of a single computation for each variable. Furthermore, the equations are local in a sense that they depend only on neurons and synapses directly connected to the updated variable, i.e. the Markov blanket of the variable. Each variable does not require any global signals such as an error of the current output of the network. Asynchronicity and locality of the update dynamics of the learning must be particularly suitable for biological implementation of the algorithm because the network needs not prepare any global signals to synchronize neural or synaptic activities, or to share a global evaluation of the network.

Note the difference between summation indexes of Eqs. (2) and (4). Synaptic update requires summation over data of a given dataset while neural updates does not. The difference is a direct outcome of the difference of the data dependency of variables that we have mentioned above. Because of the difference, synapses need to accumulate neural activities for all data of the dataset to reset their state while neurons reset their state independently for each data. This implies that, if data are provided sequentially to the network as in the brain, synapses must evolve much slower than neurons to allow the network to see all data of the dataset, which explains largely different timescales of neuron and synapse in the brain. Contrary, if we use the algorithm to train an artificial neural network, we can parallelly perform neural updates for all data of the dataset to practically accelerate leaning speed.

Synaptic plasticity

The derived update rule of synapses Eqs. (3) and (4) gives similar plasticity to cortical synapses. Each term of the summation of Eq. (3) vanishes unless the presynaptic neuron x_{di} fires, and if the neuron fires, it will be positive (LTP) or negative (LTD) depending on

whether the postsynaptic neuron x_{dj} fires at the same time. Therefore, each synaptic weight increases if the network receives many data in which the pre and post synaptic neurons of the connection fire synchronously while it decreases if the pre and post synaptic neurons fire asynchronously (Fig.1d), which agrees well with STDP of cortical synapses excepting precise spike timing that we do not model here. Interestingly, the update rule depends on membrane potential v_{dj} in addition to spikes of the postsynaptic neuron. This is consistent with a recently proposed model of STDP that explains various properties of synaptic plasticity by introducing average membrane potential of postsynaptic neurons besides spikes to the model [30]. Our result provides a theoretical foundation of the necessity of the membrane potential dependency for synaptic plasticity that empirically introduced in the previous study.

It is also noteworthy that, unlike most of existing models of synaptic plasticity, the plasticity derived here does not require any artificial bound of synaptic weights. Increment of each synaptic weight automatically saturates when the magnitude of the weight becomes very large because $\sigma(v_{dj})$ is the expectation of the x_{dj} and the difference of them, $x_{dj} - \sigma(v_{dj})$, decays to zero in such cases (Fig. 1d). The latent scalar variable of each synapse q_{ij} may correspond to size of a spine at the synaptic contact [39, 40].

Retrograde modulation of excitability

The bias term b_{di} of the neural dynamics Eq. (2) that represents retrograde modulation of firing probability of a neuron by its postsynaptic neurons provides a stochastic variant of the error propagation of the backpropagation learning. Recall that a target output is given to the network by clamping the state of corresponding visible neurons to the desired output. Assume that the jth neuron is one of these neurons, then x_{dj} gives the desired value and the term $x_{dj} - \sigma(v_{dj})$ in the bias b_{di} gives difference between the desired value and the expected value of the x_{dj} when the neuron is not clamped, which is just the same as the error of the backpropagation learning when one uses the squared error as a loss function. (Please remember that the x_{di} and v_{di} are stochastic variables when the neuron is not clamped.) The term b_{di} retrogradely accumulates errors of postsynaptic neurons to the presynaptic neuron and regulates excitability of the neuron so that it

reduces the errors. Even if the neuron is not in the output layer, the term $x_{dj} - \sigma(v_{dj})$ still represents an error of the neuron as it gives the difference between the current and the expected value of the neuron when the network does and does not receive desired outputs respectively. Owing to the retrograde modulation, information of the desired output provided only to output neurons can spread, i.e. diffuse, over the entire network even though neurons update their variables independently without coordinated scheduling of error backpropagation.

Equations (1) and (2) means that a spike of a neuron immediately changes b_{di} and excitability of presynaptic neurons. Such immediate retrograde modulation, however, has not been experimentally reported and seems biologically implausible. Rather, it is biologically more natural that b_{di} slowly evolves with accumulating effects of postsynaptic spikes as

$$b_{di} \leftarrow (1 - r_b) b_{di} + r_b \sum_j w_{ij} \left(x_{dj} - \sigma(v_{dj}) \right) \tag{5}$$

, where r_b characterizes timescale of the evolution ($0 < r_b \leq 1$ and $r_b = 1$ reproduces Eq. (2)). Here, b_{di} is determined by an average spike history of postsynaptic neurons over a finite, presumably quite long, duration. Similarly, we can also generalize evolution equation of q_{ij} to

$$q_{ij} \leftarrow (1 - r_q) q_{ij} + r_q \sum_d x_{di} \left(x_{dj} - \sigma(v_{dj}) \right). \tag{6}$$

We will numerically show later that these generalizations rarely impede learning accuracy of the algorithm.

A possible biological implementation of the algorithm can be as follows: (i) Each neuron in the network continuously evaluates its membrane potential v and the bias b and stochastically generates spikes with a probability given by Eq. (1). (ii) A generated spike is immediately integrated into membrane potential of postsynaptic neurons while it slowly modulates excitability of presynaptic neurons (Eq. (5)). (iii) The spike firing also modulates the latent synaptic variable q (Eq. (6)). (iv) Each synapse asynchronously and irregularly changes its state along Eq. (3) with frequency that is slow enough so that sensory neurons receive varieties of external inputs during the average interval of the updates.

Feedforward networks

To see how the algorithm works, we first apply the algorithm to a simple problem of supervised learning for a three-layered network (see Methods for full details). Figure 1e shows evolution of training and test accuracies as functions of sampling iteration. Accuracies almost completely overlap each other and promptly increase to values close to unity and stay there. Significantly, even while these accuracies keep almost constant values, synaptic weights of the network continue to largely fluctuate (Fig. 1f) and firing pattern of hidden neurons also keeps changing without converging to a fixed pattern (Fig. 1g). These results are consistent with experimental observations of continuing fluctuation of synapses and trial-to-trial variability of neural activity of cortex.

Arbitral constants included in the original algorithm (1)-(4) are only the number of synapses per connection K and the maximum amplitude of synapses a_0. To study robustness of the algorithm to these constants, we calculate achieved test accuracies of the network after learning for various combinations of values of them (Fig. 1h). Excepting a narrow range where one of them is so small that the possible maximum weight given by $a_0 K$ is small, the network almost perfectly learns the task, which implies that the algorithm does not require fine tuning of parameters to solve the task.

We then evaluate influence of additional parameters r_b and r_q (Figure 1i, 1j). Unless r_q is exceptionally small, namely q evolves extremely slowly, training and test accuracies successfully increase and reach to around unity whereas convergence speed decreases as we decrease r_b or r_q. (Note that the synaptic evolution is already slower than that of neurons even while $r_q = 1$ as we have discussed above. Introduction of a small r_q may result in unrealistic excess slowdown to synaptic dynamics.) We can conclude, therefore, that the learning is not practically hindered by introduction of finite time scales of evolution of b and q.

We next apply the algorithm to train multilayered feedforward networks on the MNIST dataset to demonstrate applicability of the method to a practical problem (Fig. 2). Accuracies promptly increase to values around 95%, while the number of required iterations and finally achieved accuracies slightly decreases as we increase the number of

layers in the network, (Fig. 2a-c). Figure 2d gives examples of digits a network fails to recognize, which are often ambiguous even for us to confidently recognize.

Figure 2e illustrates examples of connection weights from the input to the hidden neurons in the three-layered network after training, which correspond to receptive fields of the hidden neurons. We can see that Gabor-filter like localized structures that resembles receptive fields of neurons in the primary visual cortex are often organized through the learning. Introduction of a sparse constraint of neural activity to loss function of learning is known to provides Gabor-filter like receptive fields to an artificial neural network [41]. However, it is interesting that the network can acquire similar localized structures of receptive fields even without any explicit additional constraint of the learning algorithm.

The learning algorithm may avoid overfitting to training data because it is not derived as a direct optimization of any objective functions. To check this, we train the three-layered feedforward network using small numbers of samples of the MNIST dataset, and compare training and test accuracies of them with those obtained by the backpropagation learning using the stochastic gradient descent (SGD) and the ADAM [42] (Fig. 2f). Training accuracies of the backpropagation learning keep much higher values than test accuracies of them and decrease or stick to unity while test accuracies monotonically increase, which implies overfitting to the training dataset. To the contrary, training and test accuracies of the proposed algorithm increase almost together with keeping only small difference between them as the number of training data increases. Thus, the overfitting to training data is surely suppressed in the stochastic learning algorithm.

Most efficient power-low coding

A recent experiment of simultaneous recording of huge numbers of neurons revealed that variance spectrum of principle component of neural activities measured as two-photon calcium imaging obeys a power-law with exponent that is almost the same but slightly decays faster than the critical exponent -1 [7]. The variance spectrum that decays slower than -1 is mathematically proven to be non-differentiable while that decays faster than -1 does not fully utilize high-dimensionality of neural population. Thus, the experimentally observed power-law coding with just below the critical exponent is the most efficient in the sense that population response of neurons lies on a manifold with as high dimensional

as possible while it keeps differentiability of it, which enables robust generalization. To test whether the network trained by the derived algorithm achieves the most efficient coding with just below the critical exponent, we numerically calculate variance spectrum of principle components of the mean activity of neurons in the hidden layer of the network trained on the MNIST dataset. As shown in Fig. 2g, variance spectrum shows clear power-law decay with exponent -1.06 that is almost the same as the experimental result and just below the critical exponent -1. This indicates that the leaning algorithm leads the network to the most efficient coding, which may explain high generalization ability of the trained network that is demonstrated as almost perfectly overlap of training and test accuracies of the previous figures.

We then study how does the exponent of the power-law develops during learning. Figure 2h shows that the exponent approaches to the value close to -1 from below as the leaning proceeds. The result implies that the network first learns coarser representation of the dataset and then gradually acquires finer structures of this with keeping differentiability of representation of data in coding space rather than it starts from non-smooth almost perfect representation of training dataset and then it is organized into smooth coding to acquire robust generalization. Robustness or generalization ability of population coding, thus, seems to have priority over precision of data representation in the learning. This priority must particularly be beneficial for animals that should be survive in ceaselessly changing environment.

Recurrent networks

Then we apply the algorithm to train a network with recurrent connections (Fig. 3a) on the MNIST dataset. If we virtually unfold recurrent connections of the network along time axis, we obtain a graphical model illustrated in Fig. 3b. Strictly speaking, the Gibbs sampling should be applied on the unfolded graph, which means that we should repeat to update all variables on the graph many times. The strict application is, however, biologically and physically unrealistic because it violates causality of neural dynamics for two reasons. First, iterative update of variables means that even past variables should be repeatedly visited to be sampled even after future variables are updated. Second, updates of values of hidden neurons at a time require values of them at the next time step because they are included in the retrograde biases corresponding to the recurrent

connections. To solve the issue, we apply the algorithm to recurrent networks such that we simply set retrograde biases corresponding to recurrent connections to zero and update only present states of neurons along time development without going back to the past (Fig. 3c, Methods). This procedure recovers plausibility of the algorithm in which time goes on naturally and updates of neurons require only current and adjacent past state of them.

Figure 3d shows evolution of training and test accuracies as functions of the number of sampling iterations that is the same as the time step here. Accuracies are obtained from output of the network measured after recursive evolutions or updates of states of hidden neurons. Like the results of feedforward networks, accuracies are almost perfectly overlap each other and promptly increase, which means that the algorithm enables to train even the recurrent network with rarely showing overfitting. We also evaluate accuracies of the network with virtually eliminate the trained recurrent connections. The elimination drastically degrades achieved accuracies, which implies that the trained recurrent connections surely work positively for the classification task.

Statistics of network motifs

It is reported that local cortical circuit is highly nonrandom and connectivity patterns consisting of multiple neurons, known as network motifs, show characteristic distribution such that highly-clustered patterns are overrepresented [31]. To study whether the recurrent network trained by the algorithm acquires similar distribution of connectivity patterns, we count connectivity of triplet of neurons of a trained recurrent network. Statistics of the ratio between actual counts of triplet patterns to those of chance level are plotted in Figure 3d. Ratios of the experimental result are also overlaid in the figure. While values of the ratios largely change over trials and there are some exceptional cases in which ratios are bit higher than those of the experimental result, the distribution of triplet patterns agrees surprisingly well with that of the experimental result. As experimentally observed, highly-connected three-neuron motifs that are labeled as 10 to 16 in Fig. 3d are overrepresented several times greater than chance level. It is also interesting that even slightly higher ratio from the chance level of the pattern labeled 4 in Fig. 3d is reproduced by the model. These results may support the validity of the derived algorithm as a model to explain formation of local cortical circuit.

Temporal sequence learning

We next use the algorithm to train recurrent networks on temporal sequences, in which networks learn to store history of input for a while to generate desired output. We prepare periodic temporal sequences in which the same temporal inputs may appear multiple times at different temporal position, and train networks so that they predict the next input of the current input. For instance, if the input is $1 \rightarrow 2 \rightarrow 3 \rightarrow 2 \rightarrow 1$, target output will be $2 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 2$. Therefore, even when the network receives the same input "2", target output will be "3" or "1" depending on whether previous input was "1" or "3" respectively, which requires network memorize previous input in one step. We prepare two sequences that requires one and two steps memories respectively. Training procedure of the network is the same as that used in Fig. 3.

For both tasks, networks are successfully trained to output desired sequences by the learning algorithm. Reflecting past histories, activities of hidden neurons after learning are different between time steps when the network receives the same input if past inputs are different. This implies that, through the learning, the algorithm makes the networks can store input histories into hidden neuron's activity.

Unsupervised learning

So far, we have applied the learning algorithm to supervised learning. The algorithm is, however, also applicable to unsupervised learning that does not require any labels of input data. To use the algorithm for unsupervised learning, we employ a similar setting as an autoencoder [43]. An autoencoder is a feedforward network trained by an unsupervised learning in which the network is required to generate copy of the input via internal, typically dimensionality reduced, representation of the input. Instead of clean input data, the network may be given noisy data and required to output purified version of them, i.e. to denoise noisy input.

Figure 5 shows results of the unsupervised learning where the same inputs are simultaneously provided to neurons in the input and output layers. After learning, networks successfully learn to generate images that are similar with given inputs or denoised inputs when noisy inputs are presented. While labels indicating categories of the

inputs have not been provided to the network, activity of the hidden neurons change with correctly reflecting input category. Indeed, by fixing states of hidden neurons, we can generate images that are similar with those of the given dataset. As the possible number of generated images increases as the number of hidden neurons, the network with more hidden neurons can reconstruct finer difference of given images such as right or left tilted images of the digit “1” (Fig. 5i), which is consistent with results of the most efficient coding indicated by the critical power-law dependency of Fig. 2g

Discussion

In this study, we showed that the Gibbs sampling from a joint posterior distribution of neurons and synapses in the network conditioned on external environment, or given dataset, integrates stochastic nature of synaptic development and neural activities of cortical circuit into a unified framework, which gives a biologically plausible learning algorithm that consistently explains experimental findings of cortical circuit. In other words, stochastic spiking activity of neurons and stochastic evolution of synapses in the cortex inseparably work to realize the single primarily function of the cortex i.e. the posterior sampling. In contrast to existing learning methods, in the algorithm computations for inference and learning are not clearly separated. Apparent difference of dynamics of neurons and synapses is the outcome of the difference between data dependencies of them. The derived stochastic dynamics of synapses well explains plasticity of cortical synapses, and that of neurons naturally describes highly irregular features and trial-to-trial variability of spike trains of cortical neurons.

The evolution equation of neurons has a term of retrograde modulation of excitability of a neuron by its postsynaptic neurons. Due to the term, the algorithm works as a stochastic variant of the error backpropagation by which target outputs provided to a part of network can spread over the entire network. However, in contrast to the backpropagation learning, the retrograde modulation does not require neither synchronous nor precisely coordinated operations, which has been major obstacles to consider the backpropagation as the learning principle of the brain. The retrograde modulation need not to be immediately affected by postsynaptic action potentials but can slowly integrate postsynaptic spikes, which seems biologically plausible as it can be implemented in real cortical circuit. To our knowledge, experiments to verify existence of such slow retrograde modulation of

excitability of a neuron have not been attempted so far. Candidates of the biological implementation of this include the combination of action potential backpropagation from soma to dendritic spines [44] and retrograde transsynaptic transport of some chemicals [45], disynaptic connections from post to presynaptic neurons, and glia-mediated modulation [46]. Experimental confirmation of the modulation will be a convincing evidence to support the proposed framework as a learning and computational principle of the brain.

The reason why both neurons and synapses must be stochastic in cortex is explained as follows. Suppose that the primary purpose of the cortex is to generate synaptic weights that consistently explains external environment. Mathematically, this is formulated as to generate samples of synaptic weights from the Bayesian posterior distribution conditioned on external environment, which is derived from the likelihood function of the synaptic states given by the dataset. However, when the network has hidden neurons, the likelihood is given as a marginal distribution of the joint conditional distribution of hidden neurons and synapses where neural and synaptic variables tightly couple each other. Thus, sampling of synaptic states inevitably requires sampling of neural states together, which results in coexisting stochastic dynamics of neurons and synapses of the cortical circuit.

It has been pointed out that a temporal difference (TD) method of the reinforcement learning reproduces a spike-timing-dependent Hebbian plasticity [35]. In TD learnings, a value function of the current state is updated after reward delivery such that the value adds the TD error defined as the difference between delivered reward and its prediction in advance. Interestingly, each term of evolution equations of latent variables of synapses and neurons, i.e. b_{di} and q_{ij}, can be regarded as a representation of TD error of delivered spikes instead of rewards. Indeed, each term of Equation (2) and (4) is given as a scaled difference between a measured spike of a postsynaptic neuron x_{dj} and the expected value of this $\sigma(v_{dj})$ before observation of the spike. In this analogy, therefore, each spike plays a role of reward for presynaptic neurons and synapses from these neurons. Then, spikes that occur unexpectedly largely modulate the latent variables of them. At the same time, the spike is also anterogradely sent to postsynaptic neurons to change firing probability of them reflecting synaptic weights to them. This may imply

that each spike in cortex plays dual roles: the first is rewards for backward units and the second is an actor selecting next states simultaneously by evoking next spikes. Theoretical justification of the relationship between the algorithm and TD learning will be a fascinating future subject. Also, as posterior sampling attracts much attention recently [36], revealing relationship to this also will be an interesting future issue.

The algorithm developed here closely relates to the Boltzmann machine [34] that is a stochastic recurrent neural network where neurons are represented as stochastic binary variables and iteratively updated to achieve a thermal equilibrium of a globally defined energy function specified by a given temperature. Symmetric connection weights between neurons are trained by a gradient descent method to make the equilibrium state approximate a target distribution in which neural states of given dataset have high probability. The algorithm developed in the paper, can be considered as an extension of the Boltzmann machine in which the Bayesian posterior distribution is directly considered instead of a thermal equilibrium of an energy function and synapses besides neurons are modeled as stochastic variables that should be sampled. Owing to the extension, the algorithm can equate training of a network to inference on it, which results in the biologically realistic update rules of variables performed concurrently for neurons and synapses without explicit switching of different computations nor fine scheduling of parameters such as the simulated annealing.

In the limit of large numbers of training data, the posterior distribution of Bayesian inference generally converges to a delta function whose peak position coincides with the result of maximum likelihood estimation of model parameters under the given dataset (Methods). Therefore, if enough external data is provided to the network, the learning algorithm almost surely generates synaptic and neural variables that most suitably explains the external data. It is in contrast with the backpropagation learning in which convergence to an optimal solution is not always guaranteed even when we can use sufficiently large data.

As it seems to provide a basic principle of neural computation and learning, our model may provide a theoretical base for various findings of cortical dynamics and various methods of machine learning. Recent experimental findings of neuroscience have not

been fully utilized in recent development of machine learning, which is presumably because the backpropagation learning was not consistent with the brain. These experimental results include short-term plasticity of synapses [47], Dale's principle, long tail distribution of synaptic weight [31], columnar structure, laminar organization, canonical circuit [48, 49], and innate structure formed through developmental stages [50]. Incorporate these features to the model to clarify functional roles of them will be a fascinating future direction. Introducing precise continuous dynamics of membrane potential and spike generation mechanism to the model is also an important future direction.

References

[1] Shadlen, M. N., & Newsome, W. T. (1994). Noise, neural codes and cortical organization. Current Opinion in Neurobiology, 4(4), 569–579.

[2] Arieli, A., Sterkin, A., Grinvald, A., & Aertsen, A. (1996). Dynamics of ongoing activity: Explanation of the large variability in evoked cortical responses. Science, 273(5283), 1868–1871.

[3] Kenet, T., Bibitchkov, D., Tsodyks, M., Grinvald, A., & Arieli, A. (2003). Spontaneously emerging cortical representations of visual attributes. Nature, 425(6961), 954–956.

[4] Destexhe, A., Rudolph, M., & Paré, D. (2003). The high-conductance state of neocortical neurons in vivo. Nature Reviews Neuroscience, 4(9), 739–751.

[5] Berkes, P., Orbán, G., Lengyel, M., & Fiser, J. (2011). Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. Science, 331(6013), 83–87.

[6] Pouget, A., Beck, J. M., Ma, W. J., & Latham, P. E. (2013). Probabilistic brains: knowns and unknowns. Nature Neuroscience, 16(9), 1170–1178.

[7] Stringer, C., Pachitariu, M., Steinmetz, N., Carandini, M., & Harris, K. D. (2019). High-dimensional geometry of population responses in visual cortex. Nature, 571(7765), 361–365.

[8] Llera-Montero, M., Sacramento, J., & Costa, R. P. (2019). Computational roles of plastic probabilistic synapses. Current Opinion in Neurobiology, 54, 90–97.

[9] Stettler, D. D., Yamahachi, H., Li, W., Denk, W., & Gilbert, C. D. (2006). Axons and Synaptic Boutons Are Highly Dynamic in Adult Visual Cortex. Neuron, 49(6), 877–887.

[10] Yasumatsu, N., Matsuzaki, M., Miyazaki, T., Noguchi, J., & Kasai, H. (2008). Principles of long-term dynamics of dendritic spines. The Journal of Neuroscience, 28(50), 13592–13608.

[11] Holtmaat, A., & Svoboda, K. (2009). Experience-dependent structural synaptic plasticity in the mammalian brain. Nature Reviews Neuroscience, 10(9), 647–658.

[12] Chen, X., Leischner, U., Rochefort, N. L., Nelken, I., & Konnerth, A. (2011). Functional mapping of single spines in cortical neurons in vivo. Nature, 475(7357), 501–505.

[13] Attardo, A., Fitzgerald, J. E., & Schnitzer, M. J. (2015). Impermanence of dendritic spines in live adult CA1 hippocampus. Nature, 523(7562), 592–596.

[14] Mongillo, G., Rumpel, S., & Loewenstein, Y. (2017). Intrinsic volatility of synaptic connections - a challenge to the synaptic trace theory of memory. Current Opinion in Neurobiology, 46, 7–13.

[15] Pascoli, V., Hiver, A., Van Zessen, R., Loureiro, M., Achargui, R., Harada, M., et al. (2018). Stochastic synaptic plasticity underlying compulsion in a model of addiction. Nature, 564(7736), 366–371.

[16] Allen, C., & Stevens, C. F. (1994). An evaluation of causes for unreliability of synaptic transmission. Proceedings of the National Academy of Sciences, 91(22), 10380-10383.

[17] Katz, L. C., & Shatz, C. J. (1996). Synaptic Activity and the Construction of Cortical Circuits. Science, 274(5290), 1133–1138.

[18] Branco, T., & Staras, K. (2009). The probability of neurotransmitter release: variability and feedback control at single synapses. Nature Reviews Neuroscience, 10(5), 373–383.

[19] Deneve, S., Latham, P. E., & Pouget, A. (2001). Efficient computation and cue integration with noisy population codes. Nature Neuroscience, 4(8), 826–831.

[20] Ma, W. J., Beck, J. M., Latham, P. E., & Pouget, A. (2006). Bayesian inference with probabilistic population codes. Nature Neuroscience, 9(11), 1432–1438.

[21] Doya, K., Ishii, S., Pouget, A., & Rao, R. P. (Eds.). (2007). Bayesian brain: Probabilistic approaches to neural coding. MIT press.

[22] Soltani, A., & Wang, X.-J. (2010). Synaptic computation underlying probabilistic inference. Nature Neuroscience, 13(1), 112–119.

[23] Kappel, D., Habenschuss, S., Legenstein, R., & Maass, W. (2015). Network

Plasticity as Bayesian Inference. PLoS Computational Biology, 11(11), e1004485.

[24] Aitchison, L., & Latham, P. E. (2015). Synaptic sampling: A connection between PSP variability and uncertainty explains neurophysiological observations. arXiv preprint arXiv:1505.04544.

[25] Orbán, G., Berkes, P., Fiser, J., & Lengyel, M. (2016). Neural Variability and Sampling-Based Probabilistic Representations in the Visual Cortex. Neuron, 92(2), 530–543.

[26] Neftci, E. O., Pedroni, B. U., Joshi, S., Al-Shedivat, M., & Cauwenberghs, G. (2016). Stochastic Synapses Enable Efficient Brain-Inspired Learning Machines. Frontiers in Neuroscience, 10(99), 796.

[27] Hiratani, N., & Fukai, T. (2018). Redundancy in synaptic connections enables neurons to learn optimally. Proceedings of the National Academy of Sciences of the United States of America, 115(29), E6871–E6879.

[28] Baldassi, C., Gerace, F., Kappen, H. J., Lucibello, C., Saglietti, L., Tartaglione, E., & Zecchina, R. (2018). Role of Synaptic Stochasticity in Training Low-Precision Neural Networks. Physical Review Letters, 120(26), 268103.

[29] Bi, G. Q., & Poo, M. M. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. The Journal of Neuroscience, 18(24), 10464–10472.

[30] Clopath, C., Büsing, L., Vasilaki, E., & Gerstner, W. (2010). Connectivity reflects coding: a model of voltage-based STDP with homeostasis. Nature Neuroscience, 13(3), 344–352.

[31] Song, S., Sjöström, P. J., Reigl, M., Nelson, S. B., & Chklovskii, D. B. (2005). Highly nonrandom features of synaptic connectivity in local cortical circuits. PLoS Biology, 3(3), e68.

[32] Bengio, Y., Lee, D.-H., Bornschein, J., Mesnard, T., & Lin, Z. (2015). Towards Biologically Plausible Deep Learning. arXiv.org.

[33] Lillicrap, T. P., Cownden, D., Tweed, D. B., & Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. Nature Communications, 7, ncomms13276.

[34] Ackley, D., Hinton, G., and Sejnowski, T. (1985). A Learning Algorithm for Boltzmann Machines. Cognitive Science, 9(1):147-169.

[35] Rao, R. P. N., & Sejnowski, T. J. (2001). Spike-timing-dependent Hebbian plasticity

as temporal difference learning. Neural Computation, 13(10), 2221–2237.
[36] Osband, I., & Van Roy, B. (2016). Why is Posterior Sampling Better than Optimism for Reinforcement Learning? arXiv.org.
[37] Feldmeyer, D., Egger, V., Lübke, J., & Sakmann, B. (1999). Reliable synaptic connections between pairs of excitatory layer 4 neurones within a single “barrel” of developing rat somatosensory cortex. The Journal of Physiology, 521(1), 169–190.
[38] Geman, S.; Geman, D. (1984). "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". IEEE Transactions on Pattern Analysis and Machine Intelligence. 6 (6): 721–741.
[39] Noguchi, J., Nagaoka, A., Watanabe, S., Davies, G. C. R. E., Kitamura, K., Kano, M., et al. (2011). In vivo two□photon uncaging of glutamate revealing the structure–function relationships of dendritic spines in the neocortex of adult mice. The Journal of Physiology, 589(10), 2447–2457.
[40] Berry, K. P., & Nedivi, E. (2017). Spine Dynamics: Are They All the Same? Neuron, 96(1), 43–55.
[41] Olshausen, B. A., & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature, 381(6583), 607–609.
[42] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
[43] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. Science, 313(5786), 504–507.
[44] Stuart, G., Spruston, N., Sakmann, B., & Häusser, M. (1997). Action potential initiation and backpropagation in neurons of the mammalian CNS. Trends in Neurosciences, 20(3), 125–131.
[45] Regehr, W. G., Carey, M. R., & Best, A. R. (2009). Activity-Dependent Regulation of Synapses by Retrograde Messengers. Neuron, 63(2), 154–170.
[46] Fields, R. D., & Stevens-Graham, B. (2002). New insights into neuron-glia communication. Science, 298(5593), 556-562.
[47] Markram, H. and Tsodyks, M. (1996). Redistribution of synaptic efficacy between neocortical pyramidal neurons. Nature. 382(6594): 807-810.
[48] Silberberg, G., Grillner, S., LeBeau, F. E. N., Maex, R., & Markram, H. (2005). Synaptic pathways in neural microcircuits. Trends in Neurosciences, 28(10), 541–551.
[49] Bastos, A. M., Usrey, W. M., Adams, R. A., Mangun, G. R., Fries, P., & Friston, K. J.

(2012). Canonical Microcircuits for Predictive Coding. Neuron, 76(4), 695–711.
[50] Ullman, S. (2019). Using neuroscience to develop artificial intelligence. Science, 363(6428), 692–693.

Figure captions

Figure 1

Learning as a Gibbs sampling of synapses and neurons. (a) A neural network is modeled as population of neurons connected each other via multiple synapses. (b) A simple neural network consists of three neurons and four synapses per connection. The input, hidden, and output neurons are denoted as x_0, x_1 and x_2 respectively. (c) A graphical model representation of the network when a dataset of D data is given to the network. White and gray circles indicate free and clumped variables respectively. (d) Evolutions of a synaptic weight (right panel) when presynaptic and postsynaptic neurons fire (left panel) synchronously (cyan) and asynchronously (magenta). (e) Training (cyan) and test (magenta) accuracies of a three-layered network as functions of sampling iteration (details in Methods). (f) Evolution of randomly chosen synaptic weights of the network. Different colors are used for different presynaptic neurons. (g) State evolution of randomly chosen hidden neurons. (h) Achieved test accuracies by the learning algorithm with various values of the number of synapses per connection K and maximum amplitude of a synapse a0. (i) Evolution of accuracies for $r_b = 1.0, 0.1, 0.01, 0.001$, from above to below. (j) The same as (i) but for $r_q = 1.0, 0.1, 0.01, 0.001$, from above to below.

Figure 2

Supervised learning of feed-forward networks on MNIST dataset. (a-c) Training (cyan) and test (magenta) accuracies as functions of sampling iteration for (a) three-, (b) four-, and (c) five-layered networks. Each number in a circle represents the number of neurons in the layer. (d) Example of training (upper row) and test (lower low) images that the three-layered network fails to recognize. (e) Examples of receptive fields of hidden neurons of the three-layered network. (f) Achieved accuracies of the three-layered network as function of the size of training dataset. The network is trained by the proposed algorithm (left panel), backpropagation learning with naïve stochastic gradient descendent (middle panel) and with ADAM (right panel). (g) Variance spectrum of

principle components of the mean activity of hidden neurons of the three-layered network after training (cyan). Variances are arranged in descending order. The straight line (magenta) indicates the critical slope of the exponent -1. (h) Evolution of the power low exponent of variance spectrum as a function of sampling iteration.

Figure 3

Supervised learning of a recurrent network on the MNIST dataset. (a) A recurrent neural network (b) Graphical model of the network after we virtually unfold recurrent connections through time. (c) States of hidden neurons at a time t depend only on their previous values and states of input and output neurons at the same time t. (d) Evolution of training (cyan) and test (magenta) accuracies of the recurrent network. Blue and green lines are training and test accuracies respectively when trained recurrent connections are virtually eliminated. (e) Average ratio of actual counts of three-neuron patterns in the trained recurrent network to that predicted by the null hypothesis (gray bars, details in Methods). Error bars are $\pm 2\sigma$. Green circles are experimental results of real cortical circuit [Song].

Figure 4

Supervised learning of recurrent networks on temporal sequences. (a) Network structure. (b) Input (upper panels) and target output (lower panels) sequences of the first dataset. Asterisks indicate examples of time points where the network needs to output different patterns while current input to the network is the same. (c) Activities of hidden (upper panels) and output (lower panels) neurons after 0, 40, and 160 sampling iterations. (d) Input (upper panels) and target output (lower panels) sequences of the second dataset. (e) The same as (c) but for the second dataset.

Figure 5

Unsupervised learning of feedforward neural networks with a bottleneck layer. (a) A feedforward neural network with one hidden neuron. (b) Examples of input (upper panels) and corresponding reconstructed images after learning (lower panels) (c) The mean activity of the hidden neuron. The vertical axis is the index of input data. Input data of $1 \leq d \leq 1000$ are digit 0 while the rest are digit 1. (d) Output images when the state of the hidden neuron is fixed to 0 (left) and 1(right). (e-g) The same as (b-d) but for noisy

input images. (h-k) The same as (a-d) but for a network with a hidden layer of three neurons.

Methods

Neural networks

The neural network consists of N neurons connected each other via K synapses. Connection weight from the ith to the jth neuron is given as $w_{ij} = \sum_{k=1}^{K} s_{ijk} a_{ijk}$, where $s_{ijk} = \{0,1\}$ is the binary random variable describing the state of the kth synapse of the connection. The weights are generally asymmetric and set $w_{ii} = 0$ to avoid self-connections. Strength, or amplitude, of a synapse denoted as a_{ijk} is a constant that we do not change during learning. We use, for simplicity, an evenly spaced sequence from $-a_0$ to a_0 for values the amplitude through the paper, $a_{ijk} = a_k = (2(k-1)/K - 1)a_0$ $(k = 1,2,\cdots,K)$. The ith neuron receives inputs from presynaptic neurons of the network and randomly generates a spike with the probability $P(x_i = 1) = \sigma(v_i) = \sigma\left(\sum_j x_j w_{ij}\right)$, where the state of the neuron $x_i = \{0,1\}$ is the random binary variable representing a spike firing of the neuron. The weighted sum of inputs to the neuron $\mathrm{v_i}$ may correspond to membrane potential of the neuron. The sigmoidal function $\sigma(x) = (1 + e^{-x})^{-x}$ is used as an activation function of a neuron.

Neurons in the network are classified into three groups, input, output, and hidden neurons. Input and output neurons together are referred to as visible neurons. External data including target outputs of supervised learning are given to the network by clamping states of visible neurons to values of the data. Each data, therefore, must be a binary vector. When we obtain output of the network after and during learning, we clump only input neurons with keeping output and hidden neurons free.

States of neurons including hidden and visible neurons are generally different when different data of a dataset is given to the network. Contrary, that of synapses are shared by all data of the dataset because the purpose of learning is to obtain a network, i.e. a set of synaptic states, that consistently explain all of data in the dataset. We, therefore, denote the state of ith neuron when the network receiving the dth data of the dataset as x_{di} using the data index while the state of the synapse s_{ijk} does not have the data index. The number of variables of the network is thus $ND + N(N-1)K$. Figure 1c shows a

graphical model representation of data dependency of variables for the simplest case of a three-layered neural network.

Learning algorithm

Learning of the network is modeled as a Gibbs sampling of all free variables from the posterior joint distribution of them conditioned on clumped variables $P\left(\{x_{di}\}_{i\in H,d\in D},\{s_{ijk}\}_{i,j\in N,k\in K}\middle|\{x_{di}\}_{i\in V}\right)$. The Gibbs sampling allows us to replace sampling from the generally high-dimensional joint distribution with a repetition of samplings of each single variable from a posterior distribution conditioned on all other variables;

$$P\left(x_{di}\middle|\{x_{dj}\}_{j\neq i},\{s_{ijk}\}\right)$$

and

$$P\left(s_{ijk}\middle|\{x_{di}\},\{s_{lmn}\}_{lmn\neq ijk}\right)$$

for a neuron and a synapse respectively. In the Gibbs sampling, the order of samplings needs not to be fixed but can be random. Also, sampling frequencies of different variables can be different. Therefore, each synapse and neuron in the network can nonperiodically changes its state in arbitral frequency regardless of when other variables change their states in principle.

To derive an explicit description of the posterior distribution of a neuron, let us consider the log likelihood ratio of x_{di}. Using the Bayes rule, we obtain

$$\begin{aligned}
&\log\frac{P(x_{di}=1|\cdots)}{1-P(x_{di}=1|\cdots)}=\log\frac{P(x_{di}=1|\cdots)}{P(x_{di}=0|\cdots)}\\
&=\log\frac{P(x_{di}=1)}{P(x_{di}=0)}\prod_j\frac{P\left(x_{dj}\middle|x_{di}=0,\cdots\right)}{P\left(x_{dj}\middle|x_{di}=1,\cdots\right)}\\
&=\frac{\sigma(v_{di})}{1-\sigma(v_{di})}\\
&+\sum_j\begin{cases}\log\sigma\left(v_{dj,-i}+w_{ij}\right)-\log\sigma\left(v_{dj,-i}\right), & x_{dj}=1\\ \log\left(1-\sigma\left(v_{dj,-i}+w_{ij}\right)\right)-\log\left(1-\sigma\left(v_{dj,-i}\right)\right), & x_{dj}=0\end{cases}
\end{aligned}$$

$$= v_{dj} + \sum_j \begin{cases} \left(1 - \sigma\left(v_{dj}\right)\right) w_{ij}, & x_{dj} = 1 \\ -\sigma\left(v_{dj}\right) w_{ij}, & x_{dj} = 0 \end{cases}$$

$$= v_{dj} + \sum_j \left(x_{dj} - \sigma\left(v_{dj}\right)\right) w_{ij}$$

, where "$\cdots$" abbreviates all variables excepting x_{di}, i.e. $\left\{x_{dj}\right\}_{j \neq i}$ and $\left\{s_{ijk}\right\}$, and $v_{dj,-i} = \sum_{k \neq i} x_{dk} w_{kj}$. To obtain the 5th line, we assume that $v_{dj,-i} \gg w_{ij}$ and linearize each term of the summation to w_{ij}. Solving the above equation for $p(x_{di} = 1 | \cdots)$, we obtain Eqs. (1) and (2) of the main text as the posterior distribution, i.e. the stochastic update rule, of the neuron.

Similarly, log likelihood ratio of the synapse s_{ijk} follows

$$\log \frac{P\left(s_{ijk} = 1 \middle| \cdots\right)}{1 - P\left(s_{ijk} = 1 \middle| \cdots\right)} = \log \frac{P\left(s_{ijk} = 1 \middle| \cdots\right)}{P\left(s_{ijk} = 0 \middle| \cdots\right)}$$

$$= \log \frac{P\left(s_{ijk} = 1\right)}{P\left(s_{ijk} = 0\right)} \prod_d \frac{P\left(x_{dj} \middle| s_{ijk} = 0, \cdots\right)}{P\left(x_{dj} \middle| s_{ijk} = 1, \cdots\right)}$$

$$= q_{0,ijk}$$

$$+ \sum_d \begin{cases} \log \sigma\left(v_{dj,-ik} + x_{di} a_{ijk}\right) - \log \sigma\left(v_{dj,-ik}\right), & x_{dj} = 1 \\ \log\left(1 - \sigma\left(v_{dj,-ik} + s_{di} a_{ijk}\right)\right) - \log\left(1 - \sigma\left(v_{dj,-ik}\right)\right), & x_{dj} = 0 \end{cases}$$

$$= q_{0,ijk} + \sum_d \begin{cases} \left(1 - \sigma\left(v_{dj}\right)\right) x_{di} a_{ijk}, & x_{dj} = 1 \\ -\sigma\left(v_{dj}\right) x_{di} a_{ijk}, & x_{dj} = 0 \end{cases}$$

$$= q_{0,ijk} + a_{ijk} \sum_j x_{di} \left(x_{dj} - \sigma\left(v_{dj}\right)\right)$$

, where "$\cdots$" abbreviates $\{x_{di}\}$ and $\{s_{lmn}\}_{lmn \neq ijk}$, and $v_{dj,-ik} = \sum_l x_l w_{lj} - x_{di} s_{ijk} a_{ijk}$. To obtain the 5th line, we assume $v_{dj,-ik} \gg a_{ijk}$, and approximate each term of the summation as a linear term of a_{ijk}. A constant $q_{0,ijk}$ represents log likelihood ratio of the prior distribution $P\left(s_{ijk}\right)$, which simply vanishes unless the prior distribution is biased. We assume $q_{0,ijk} = 0$ through the paper. Solving the above equation for $P\left(s_{ijk} = 1 \middle| \cdots\right)$ gives Eqs. (3) and (4) that are the explicit description of the posterior distribution or the update rule of the synapse.

Numerical simulations

All numerical simulations are written in python with an open-source matrix liberally Cupy. Detailed procedures to train a feedforward network are as follows: (i) We first prepare binary variables of ND neurons and WK synapses where N is the number of neurons, D is the number of data of a training dataset, W is the number of connections of the network, and K is the number of synapses per connection. (ii) Then we clump variables of visible neurons to values of data of the training dataset, and initialize values of hidden neurons and synapses randomly to 0 or 1 with the probability of 1/2. (iii) To avoid perfectly synchronized updates, we randomly choose the ratio r_x of hidden neurons and update their variables according to Eqs. (1) and (5). (iv) Similarly, we randomly chose the ratio r_s of WK synapses to update them by following Eqs. (2) and (6). (iv) We repeat (iii) and (iv) as many times as we want. Procedures to obtain prediction of the network are the same with that of the training procedure excepting that we clump only input neurons and update hidden and output neurons by Eqs. (1) and (5) with keeping synaptic values are fixed. Training and test accuracies are defined as the ratio of the number of inputs that enables the network to generate collect outputs to the total numbers of the inputs of the training and test dataset.

Training procedure for a recurrent network is as follows. Let us denote input and target vectors for the recurrent network at time t as $x_0(t)$ and $x_2(t)$, the state vector of hidden neurons at time t as $x_1(t)$, and the set of synaptic states from the ith to the jth layers as s_{ij}. Dimension of vectors $x_0(t)$, $x_1(t)$, $x_2(t)$ are N_0D, N_1D, and N_2D, respectively where N_i is the number of neurons in the ith layer. Membrane potential of hidden neurons at time t is denoted as $v_1(t)$ and retrograde biases of hidden neurons from the output neurons at time t is denoted as $b_{12}(t)$. Using the notations, the procedure is given as the follows process. (i) We first randomly initialize $x_1(0)$, s_{01}, s_{11}, s_{12}, and set $t = 1$. (ii) Then, we calculate $v_1(t)$ and $b_{12}(t)$ from $x_0(t)$, $x_2(t)$ and $x_1(t-1)$, and (iii) sample $x_1(t)$ using $v_1(t)$ and $b_{12}(t)$, then (iv) update s_{01} using $x_0(t)$ and $x_1(t)$, s_{11} using $x_0(t-1)$ and $x_1(t)$, and s_{12} using $x_1(t)$ and $x_2(t)$. (v) Repeat (ii)-(iv) with proceeding time step $t \to t+1$. Note that in the algorithm, updates of variables depend only on current and the next preceding variables and do not violate causality.

Dataset

Excepting Figs. 1 and 4, we use MNIST dataset that consists of a training dataset of 60000 examples and a test dataset of 10000 examples in which each image has 28×28 pixels. Since pixels of MNIST data range from 0 to 255, we replace them to 0 or 1 depending on whether the pixel is below or above $255/2$ to obtain binary input vectors of 784 dimension.

In Fig. 1, we train a three-layered network consisting of 40 input, 40 hidden, and 2 output neurons on a simple task that is a noisy and high dimensional variant of the XOR problem. Datasets are artificially generated as follows. We first prepare two dimensional binary vectors (X_{d1}, X_{d2}) where d is the data index of the dataset. Then to obtain 40-dimensional binary input vectors $(Y_{d1}, \cdots, Y_{d40})$, we set $Y_{di} = X_{d1}$ for $i = 1, \cdots, 20$ and $Y_{di} = X_{d2}$ for $i = 21, \cdots, 40$, and then flip the values of them randomly with a probability of 0.1 to obtained randomly scattered input dataset. Desired outputs of two dimensional vectors are given as Z_{di} =(0,1) if $\mathrm{XOR}(X_{d1}, X_{d2}) = 0$ and $(0, 1)$ if $\mathrm{XOR}(X_{d1}, X_{d2}) = 1$. The number of examples of both the training and the test dataset is 400.

In Fig. 4, we use datasets of temporal sequences to train recurrent networks. Let us denote input data and desired outputs at time t as $X_{di}(t)$ and $Z_{di}(t)$ that are fed to the network by clumping neurons in the input layer as $x_{0,di}(t) = X_{di}(t)$, $0 < i \leq N_0$ and those of output layer as $x_{2,di}(t) = Z_{di}(t)$, $0 < i \leq N_2$ where N_0 and N_2 are numbers of neurons in the input and output layer respectively. A dataset is prepared as follows. We first prepare an integer sequence $s(t)$ where $1 \leq s(t) \leq S$ and $1 \leq t \leq T$, then set $X_i(t) = 1$ if $(s(t) - 1)N_0/S < i \leq s(t)N_0/S$ and $X_i(t) = 0$ otherwise. Target output is set as $Z_{di}(t) = X_i(t + 1)$ if $1 \leq t < T$ and $Z_{di}(T) = X_i(0)$. To obtain scattered input vectors $X_{di}(t)$, we replicate $X_i(t)$ and randomly flip them as $X_{di}(t) = X_i(t)$ with probability 0.9 and $X_{di}(t) = 1 - X_i(t)$ with probability 0.1. Integer sequences $s(t)$ of $1, 2, 3, 4, 5, 4, 3, 2, 1, 2, 3, 4, 5, 4, 3, 2$ with $N_0 = 250, S = 5, T = 16$ is used as the first dataset and $1, 1, 2, 2, 3, 3, 2, 2, 1, 1, 2, 2, 3, 3, 2, 2, 1, 1, 2, 2, 3, 3, 2, 2$ with $N_0 = 150, S = 3, T = 24$ is used for the second dataset.

Parameters

We used $r_b = r_q = r_x = 1.0$, $r_s = 0.001$, $m_0 = m_1 = 10$, $K = 50$, $a_0 = 0.5$, and $D = 400$ for Fig. 1 but $K = 200$, $a_0 = 0.1$, and $D = 100$ for Fig. 1d. For all other figures, we used $r_b = 0.01$, $r_x = 0.9$, $r_q = 0.1$, $t_s = 0.001$, $m_0 = m_1 = 20$, $K = 100$, and $a_0 = 0.1$ excepting $K = 200$ for Figs. 2g, 2h and 5, $r_x = 0.5$ for Fig. 3, and $r_b = 0.1$, $r_x = 1$, $r_q = 0.05$, $r_s = 0.01$ for Fig. 4. The number of hidden neurons of the three-layered network that are not specified in figures are 1000 for Figs. 2f, 2g, 2h, 100 for Fig. 3e, 400 for Fig. 4.

Data analysis

Spectrum variance of principle components

After we trained the three-layered neural network of Fig.2a on the MNIST dataset, we fixed synapses and obtained the average activities of hidden neurons $\{\sigma(v_{di})\}_{i\in H}$. Note that v_{di} of the hidden neurons are deterministic now because both input neurons and synaptic connection from them are fixed. Principle component analysis is applied to the average activities after they are standardized. Then we obtained explained variance of each principle component, which is the same as an eigenvalue of the covariance matrix of the standardized average activities, and align them in descending order. Exponent of the power low is estimated by least square linear fit of variance spectrum in log-log space.

Statistics of network motifs

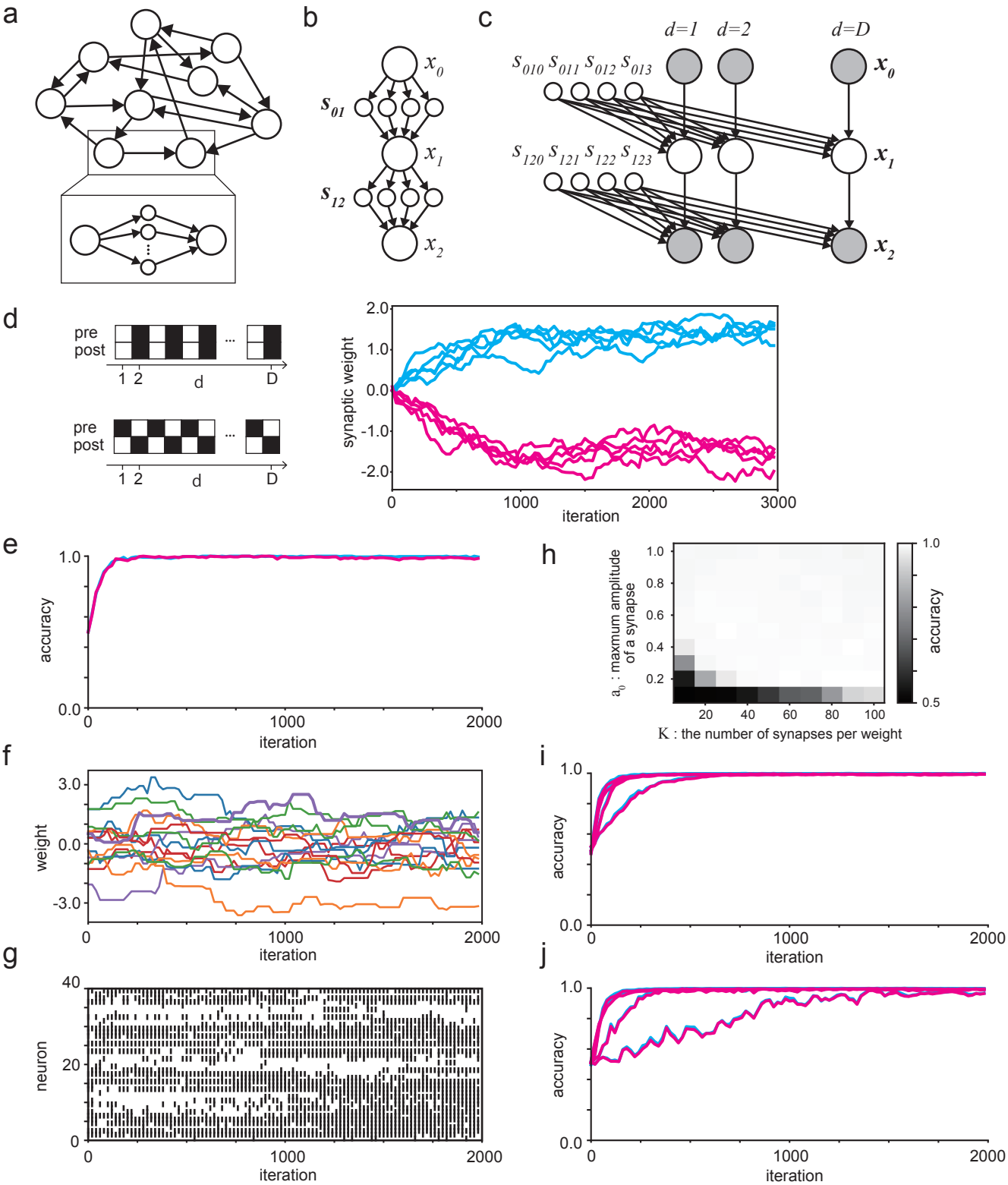
We trained a three-layered recurrent network with 100 hidden neurons. Then, we count connection patterns among triplets of neurons over all possible combinations of three of 100 neurons, i.e. for $100 \cdot 99 \cdot 98/6 = 161700$ triples. Here, we only count connections whose synaptic weight is equal or larger than 0.28 to exclude small and negative connections. Null hypothesis of the counts is defined as the same way with the paper [song] to remove effects of two-neuron patterns. We count unidirectional and bidirectional connections in all pairs of neurons and calculate the predicted count of three-neuron patterns by assuming all constituent pairs of neurons in each triplet pattern are independently chosen but with keeping probability of measured unidirectional and bidirectional connections. We performed 100 trials of learnings to obtain the mean and standard deviation σ of the ratios of actual count of each triplet pattern and that of the null hypothesis.

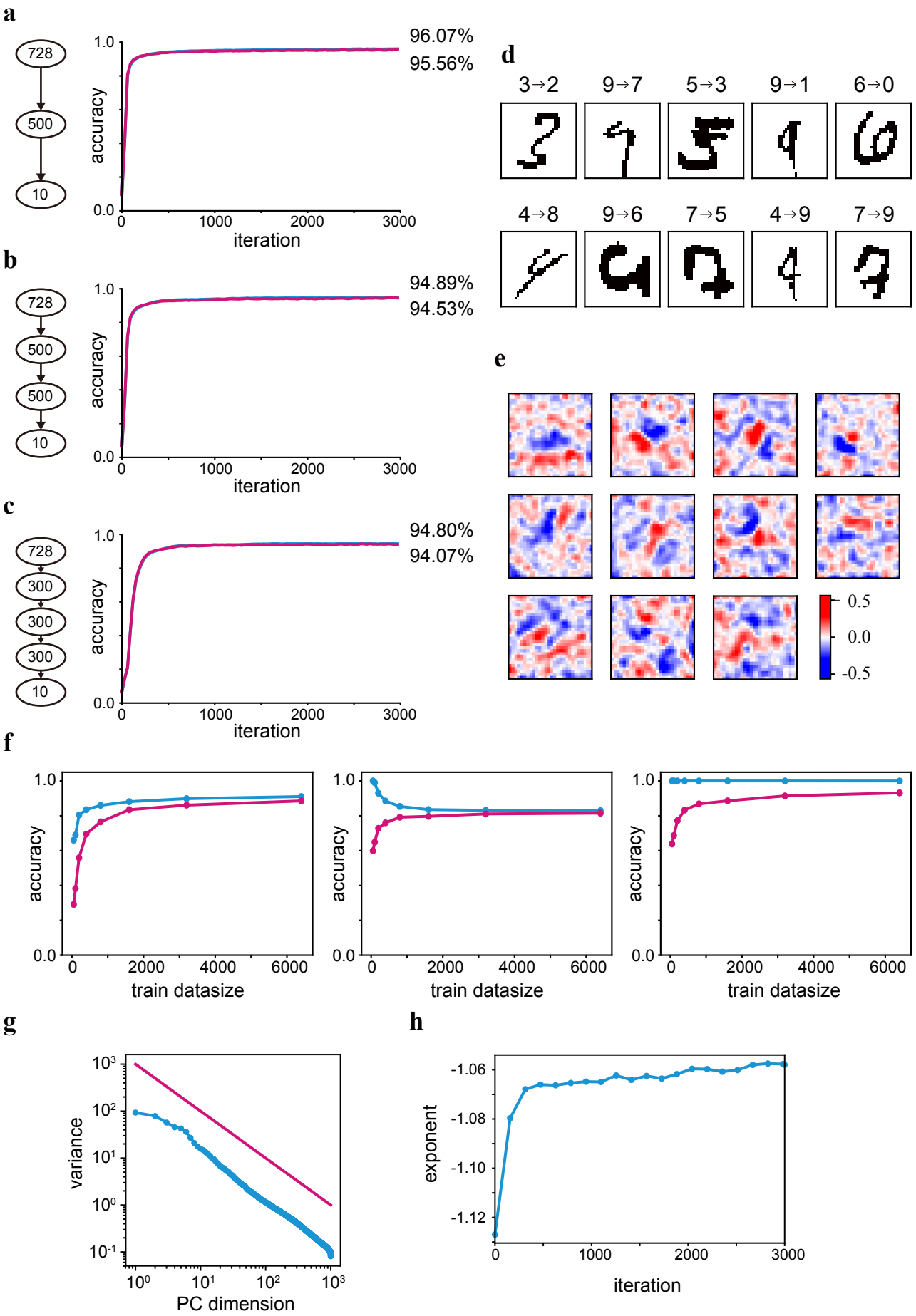
Limit of large training data size

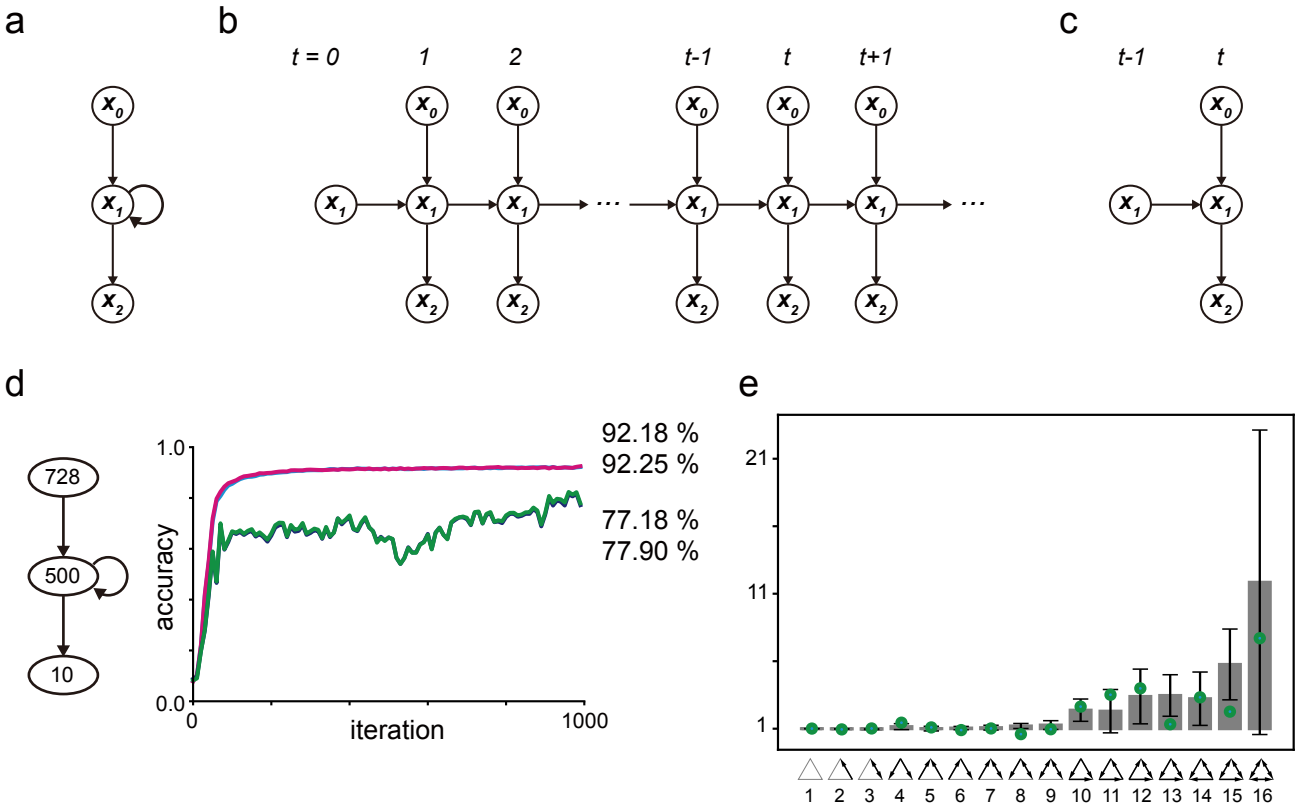
Let us denote all free variables of a network $\{x_{di}\}$ and $\{s_{ijk}\}$ together as θ. We can reasonably assume that the prior distribution $P(\theta)$ is positive for all values of θ and each training data are independently generated from a data distribution $P_d(x)$. Then, the posterior distribution satisfies

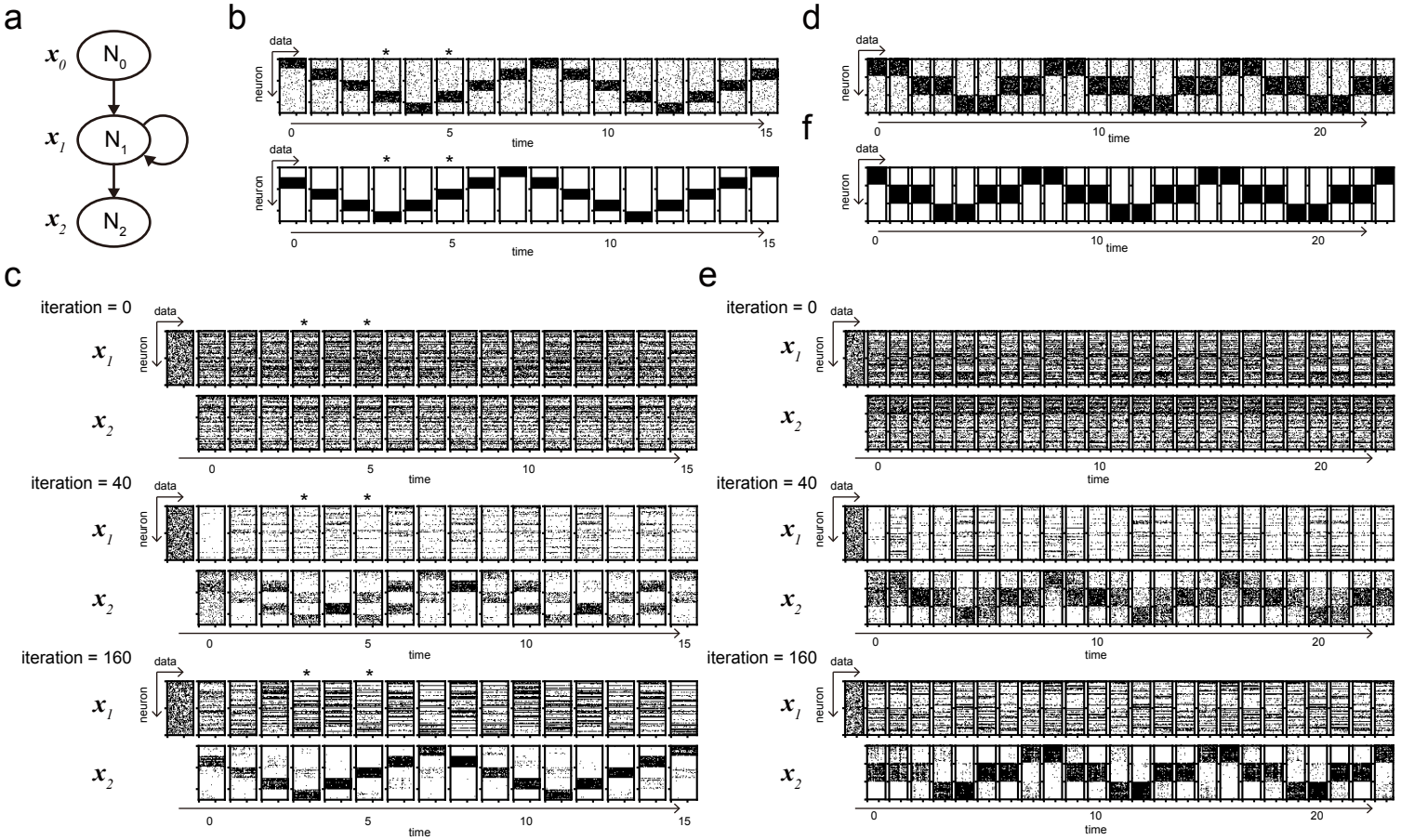
$$\begin{aligned} P(\theta|\{x_d\}) &\propto \prod_d P(x_d|\theta)P(\theta) \\ &= \exp\left(\sum_d \log P(x_d|\theta) + \log P(\theta)\right) \\ &\cong \exp\left(N_d \int \log P(x|\theta)\, P_d(x)dx + \log P(\theta)\right) \end{aligned}$$

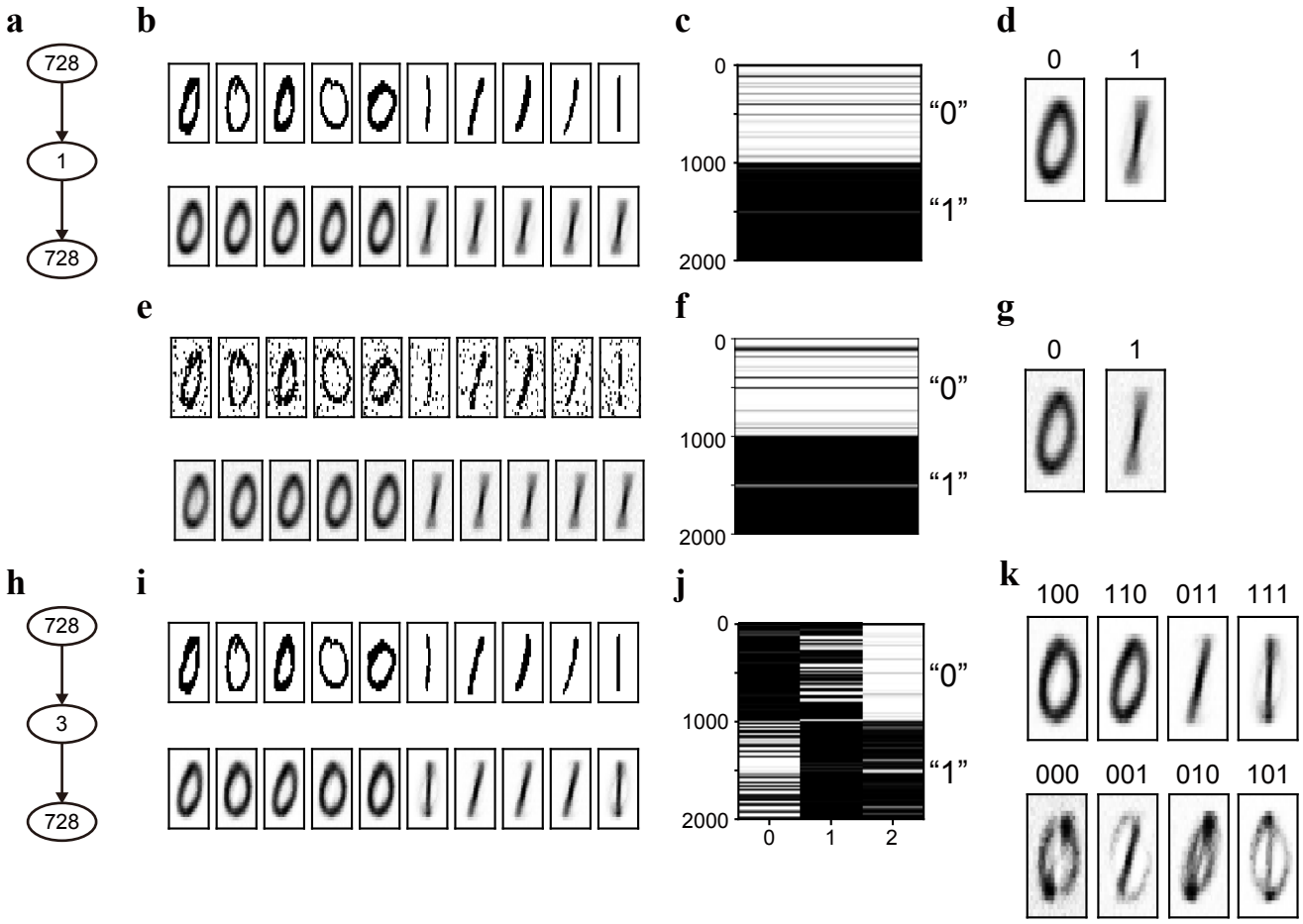
that generally converges to a delta function $\delta(\theta - \theta_0)$ in the limit of large number of the training data $N_d \to \infty$ where $\theta_0 = \mathrm{argmax}_\theta \int \log P(x|\theta)\, P_d(x)dx$. (If multiple values of θ simultaneously achieve the maximum, the posterior distribution will converge to the sum of delta functions.)


a
b
c
d
e
f
g
h
i
j
x_0
x_1
x_2
s_{01}
s_{12}
d=1
d=2
d=D
s_{010} s_{011} s_{012} s_{013}
s_{120} s_{121} s_{122} s_{123}
pre
post
1 2
d
D
synaptic weight
iteration
accuracy
weight
neuron
a_0 : maxmum amplitude of a synapse
K : the number of synapses per weight


a
728
500
10
accuracy
iteration
96.07%
95.56%
b
728
500
500
10
94.89%
94.53%
c
728
300
300
300
10
94.80%
94.07%
d
3→2
9→7
5→3
9→1
6→0
4→8
9→6
7→5
4→9
7→9
e
0.5
0.0
-0.5
f
train datasize
g
variance
PC dimension
h
exponent
iteration


a
b
c
t = 0
1
2
t-1
t
t+1
d
728
500
10
1.0
0.0
accuracy
0
iteration
1000
92.18 %
92.25 %
77.18 %
77.90 %
e
21
11
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16


a
x_0
N_0
x_1
N_1
x_2
N_2
b
data
neuron
time
0
5
10
15
d
f
20
c
e
iteration = 0
iteration = 40
iteration = 160
x_1
x_2


a
728
1
728
b
c
0
1000
2000
"0"
"1"
d
0
1
e
f
0
1000
2000
"0"
"1"
g
0
1
h
728
3
728
i
j
0
1000
2000
0
1
2
"0"
"1"
k
100
110
011
111
000
001
010
101