Natural Computing
https://doi.org/10.1007/s11047-022-09939-6

Information theoretical properties of a spiking neuron trained with Hebbian and STDP learning rules

Dominique Chu[1]

Accepted: 7 December 2022

Abstract
Using formal methods complemented by large-scale simulations we investigate information theoretical properties of spiking neurons trained using Hebbian and STDP learning rules. It is shown that weight space contains meta-stable states, which are points where the average weight change under the learning rule vanishes. These points may capture the random walker transiently. The dwell time in the vicinity of the meta-stable state is either quasi-infinite or very short and depends on the level of noise in the system. Moreover, important information theoretic quantities, such as the amount of information the neuron transmits are determined by the meta-stable state. While the Hebbian learning rule reliably leads to meta-stable states, the STDP rule tends to be unstable in the sense that for most choices of hyper-parameters the weights are not captured by meta-stable states, except for a restricted set of choices. It emerges that stochastic fluctuations play an important role in determining which meta-stable state the neuron takes. To understand this, we model the trajectory of the neuron through weight space as an inhomogeneous Markovian random walk, where the transition probabilities between states are determined by the statistics of the input signal.

Keywords Hebbian learning · Spike-timing dependent plasticity · Stochastic systems · Random walk

1 Introduction

Spiking neural networks (SNN) (Maass 1997; Gerstner and Kistler 2002) are an alternative to the more commonly used rate-coding artificial neural networks (ANN) based on McCulloch-Pitts neurons, which underpin most of modern deep learning. While ANNs are indispensable for modern machine learning, SNNs have a number of advantages that make them an interesting alternative. (*i*) It is now well known that a single spiking neuron is computationally more powerful than a perceptron (Maass 1997; Rubin et al. 2010). For example, a spiking neuron can solve the XOR problem (Fil and Chu 2020), a task on which the perceptron famously fails. (*ii*) SNNs can be implemented on neuromorphic hardware (Lin 2018) such as Davies (2018) or Plana (2011) which promise high execution speed combined with ultra-low power-consumption. (*iii*) Finally, SNNs are particularly suitable for learning of temporal data (Kreiser et al. 2017; Manette 2011).

An important class of learning rules for SNNs are Hebbian learning algorithms, in particular the classical Hebb rule (Hebb 1949; Oja 1982) and its generalisation *spike timing dependent plasticity* (STDP) (Caporale and Dan 2008; Kozdon and Bentley 2018; Lobov et al. 2020; Białas et al. 2020; Long 2011). Unlike backpropagation-based algorithms (Shrestha et al. 2018; Fang 2021) Hebbian and STDP algorithms can be implemented on neuromorphic hardware and thus benefit from its advantages. This makes them an important class of algorithms in the context of SNNs.

The general idea shared by all variations of Hebbian rules is that the connection between two neurons is strengthened (i.e. the weight increased) if the two neurons fire at the same time. In the basic Hebbian learning rule this usually means that the weight of an input channel i is increased if this channel has fired immediately before the neuron produced an output spike. STDP is an extension of this basic Hebbian rule: The weights associated with an input channel i of a neuron are increased (decreased) when

✉ Dominique Chu
d.f.chu@kent.ac.uk

[1] CEMS-School of Computing, University of Kent, Canterbury CT2 7NF, UK

Published online: 06 January 2023

i fires within a certain time window τ before (after) the neuron produces an output spike.

Given the conceptual importance of Hebbian and STDP learning both in neuroscience and in neural networks, there has been interest in understanding these learning rules theoretically. An important line of mathematical analysis focusses on understanding how the learning algorithms impact the long-term weight evolution; see for example (Kempter et al. 1999; Markram et al. 2012; Oja 1982). A common approach to this is to assume that learning is deterministic (Gerstner and Kistler 2002; Akil et al. 2020, 2021). This means that the evolution of weights is not materially impacted by random fluctuations in the input data, which simplifies the mathematics. This assumption may not always be true. Indeed, in this contribution, we will show that noise plays an essential, qualitatively important role for the dynamics of Hebbian and STDP learning. This is true even for the fully deterministic neuron model that we consider here, where the sole source of stochastic fluctuations is the noise in the data and the presentation of the data.

Our contributions are as follows: We conceptualises the neuron as performing a Markovian random walk in weight space (Leen and Moody 1992; Orr and Leen 1992) and show that the dynamics of the spiking neuron is dominated by noise, in the sense that the strengths of stochastic fluctuations determine how the weights will evolve in the long-run, and thus what the neuron learns. We then show that for the Hebbian neuron there are well identified regimes of behaviour. Within each regime, the neuron evolves towards a particular set of weights and has specific information processing properties, as measured by the mutual information between input and output. Which one of the regions the neuron "chooses" depends on the parameters, especially the amount of noise. Within each regime, the behaviour of the neuron is insensitive to variations of the parameters, but transitions between regions are discontinuous. In contrast, the STDP neuron is unstable and does not settle into meta-stable states for most choices of parameters. To the best of our knowledge, the discrete nature of the learning dynamics has not been characterised before. Our results thus provide novel insights into how the outcome of Hebbian learning algorithms is determined by the hyper-parameters of the neuron.

To derive our results, we use theoretical approaches which are then corroborated and extended by large scale numerical simulations. We will limit the scope of the investigation to a single neuron. Extending this to networks of neurons would be interesting, but would also require more space than is available for this article. Furthermore, we will focus most of our results on unstructured, random data. Prima facia, this is an unrealistic choice because real data will have a lot of structure and complicated statistical dependencies. On the other hand, there is an unlimited number of different real data sources. Even if we considered a large sample of those, no general insight could be won from what would remain a small fraction of all possible data-sets. We therefore chose synthetic, random data to understand specifically the effects of noise. That said, we will sanity check the validity of our results, when we dedicate section 4.2.1 to showing that our conclusions remain valid when a real-world data source is used.

2 Methods

In this section we will describe the methods we used to simulate the neuron. First we will introduce the neuron model. Subsequently, we describe technical details on how simulations were conducted and how we determined key indicators of the neuronal behaviour.

2.1 Neuron model

Throughout this contribution we consider only the dynamics of a single neuron with N input channels and a weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_N)$. We assume that each of the input channels i is associated with a weight $w_i \in [0, 1]$. To reduce the parameter space we enforce a strict weight-normalisation of the neurons, that is $\sum_i w_i = 1$ during initialisation and following any weight update. The internal state $V(t)$ of the neuron is initialised to $V(0) = 0$. Each time the neuron receives an input spike through channel i the internal state is increased by w_i. Throughout this contribution we assume that the neuron is updated in continuous time. A consequence of this is that it never experiences two simultaneous input spikes.

Once the internal state $V(t)$ crosses a user-determined threshold θ from below, then the neuron generates an "output spike" and the internal state is reset to 0. For our purposes here, the output spike itself will not be modelled, but simply recorded as having taken place.

Hebbian learning of the neuronal weights is coupled to output spike generation. Whenever the neuron creates an output spike, we determine the last input channel that fired before the output was generated. Calling this channel j, we then increase the value of the j-th weight by the learning rate ϵ, i.e. $w_j \leftarrow w_j + \epsilon$. Following this weight update, we then normalise the weights of all channels i to 1 by setting $w_i \leftarrow w_i / \sum_j w_j$. Altogether this results in a weight change from w_i to $(w_i + \epsilon)/(1 + \epsilon)$, for the channel whose weight is increased and in a weight change from w_j to $w_j/(1 + \epsilon)$ for all other channels. We will henceforth refer to a spiking neuron that is updated according to the Hebbian learning rule as a *Hebbian neuron*.

Similarly, for the case of the *STDP neuron* — a spiking neuron trained using the STDP rule — we increase by ϵ the weights of all channels i that fired τ time units before an output spike was generated and decrease the weights of the channels that fired within τ time units after an output spike. Weights will be normalised to one after each update. Throughout this article, we set $\tau = 0.1$.

Metastable states are points in parameter space where the effective learning rate vanishes *on average* (Chu and Nguyen 2021). This means that the expected increase and decrease of weights at this point is zero for all weights. There may be (and typically are) several metastable states in weight space. It is possible to determine conditions that the weights have to fulfil when the neuron is in a metastable state. Following (Chu and Nguyen 2021), for the Hebbian neuron this condition is simply

$$P(i|o) = w_i \tag{1}$$

where $P(i|o)$ is the conditional probability that input channel i has fired last given that the neuron has triggered a spike output.

For the STDP neuron a more complicated condition was derived in Chu and Nguyen (2021):

$$w_i = \frac{P_P \cdot P_P(i|o)(1-\epsilon) - P_D \cdot P_D(i|o)(1+\epsilon)}{P_P(1-\epsilon) - P_D(1+\epsilon)}, \tag{2}$$

where $P_P(i|o)$ is the probability that the weight of channel i is increased given that there was an output spike and $P_D(i|o)$ is the probability that the weight of channel i is decreased given that there was an output spike. Similarly, P_P and P_D are the (unconditional) probabilities that a weight will be increased or decreased respectively with $P_D + P_P = 1$.

2.2 Random input data— Poissonian neuron

We will explore the properties of the spiking neurons when they are stimulated with *independent identically distributed* (iid) random input. This means that each channel receives input spikes with a rate of μ and the inter-spike waiting times are distributed exponentially. In this case generating an input spike consists of two steps. Firstly, choose an input channel which will receive the spike by drawing a random number from the set $\{1, \ldots, N\}$, each with probability $1/N$. Secondly, generate a time interval Δt since the last spike happened, by drawing a number from an exponential distribution with parameter $\lambda = N \cdot \mu$. Throughout this paper, we set $\mu = 0.9$.

2.3 Preparation of MNIST dataset

To train the neuron on the MNIST (Lecun et al. 1998) dataset, we used a spiking neuron with $N = 28$ input channels. During each training episode, we considered only a single class of images from the dataset. Here, we arbitrarily chose images showing the digit 5. We started the training by setting the time $t = 0$. Next, we chose randomly an image of the digit 5. To simplify the computation, we limited ourselves to pixels chosen from line 14 from each image. Next we chose a random pixel in this row. The probability to choose the pixel in column i was given by $g_i / \sum_{j=1}^{28} g_j$, where g_i is the intensity of the pixel i. If the k-th pixel was chosen, then we provided an input spike to the k-th input channel of the neuron. We then updated the time to $t \leftarrow t + \Delta t$, where Δt is a random number drawn from an exponential distribution with parameter $\lambda = (28 \cdot 0.9)$. We repeated this procedure until $t > 60000$.

2.4 Determining the distance from a metastable state

Due to random fluctuations that occurred as a result of learning, the weights of a neuron will normally not be exactly in a metastable state. In order to know how far away the neuron is from a metastable state, we measure the *distance from a metastable state* or simply the "distance," following (Chu and Nguyen 2021). In order to determine the distance in the case of the Hebbian neuron, we first estimated in simulation the probability $P(i|o)$ that input channel i triggered an output spike. To do this, we stopped the learning process (i.e. weight updates) but continued to provide input. In order to estimate $P(i|o)$, we determined the relative frequency f_i with which each of the input channels triggered an output spike. Excluding all those channels where $f_i = 0$, we then calculated the distance as

$$d := \sum_i \left(1 - \frac{w_i}{f_i}\right). \tag{3}$$

In the case of the STDP learning, the distance is computed analogously, as the difference between the left hand side and the right hand side of Eq. 2. To compute this, we estimated (from simulation) the probability of the weight of any channel firing within the relevant time window before (P_P) and after (P_D) an output spike with $P_P + P_D = 1$; these values could be interpreted as the probabilities of weights being promoted/demoted as a part of the STDP algorithm. We also determined the probabilities f_i (g_i) that the i-th channel was found to fire within a relevant time window before (after) an output spike was triggered. The distance was then calculated as:

$$d := \sum_i \left(\frac{P_P \cdot f_i(1-\epsilon) - P_D \cdot g_i(1+\epsilon)}{P_P(1-\epsilon) - P_D(1+\epsilon)} - w_i\right). \tag{4}$$

3 Theory

In this section, we will present some theoretical results about the dynamics of the Hebbian and STDP neuron. The insights that we present here will then be confirmed and complemented with simulation results in Sect. 4.

3.1 Learning as a random walk

The training of the neuron can be modelled as a Markovian random walk in weight space. The walker takes a step each time the learning rule is invoked, or equivalently, each time the neuron produces an output spike. Let $P_i(w,t)$ denote the probability that after t learning episodes the weight of the i-th input channel takes the value w, where $w \in [0,1]$ and $t \in \mathbb{N}$. Then the master equation for the system reads:

$$\begin{aligned} P_i(w,t+1) =& P(i|o)P_i(w(1+\epsilon),t) \\ &+ (1-P(i|o))P_i(w+w\epsilon-\epsilon,t) \end{aligned} \tag{5}$$

Here $P(i|o)$ is the probability that given that the neuron fired an output spike the weight of channel i will be increased. In the case of the STDP algorithm, this would be the probability that channel i fired within a time τ before an output spike was triggered. The probability $P(i|o)$ depends on the current weight vector $\mathbf{w}$, not just on w_i, and hence couples the dynamics of the different weight channels. The update rule itself is local though.

In practice, the master equation Eq. 5 is difficult to solve for at least two reasons. (*i*) The functional form of $P(i|o)$ is unknown, and indeed intractable to compute, except in some special cases. (*ii*) The step size of the random walker depends on the weights. For example, if the current weight is $w(1+\epsilon)$ then the weight decrease would be by $w\epsilon$ to w. As $w \to 0$ the step size vanishes, but as $w \to 1$ the weight decrease maximises to ϵ. A similar argument applies to weight increases. The varying step size makes this an inhomogeneous random walk. This class of random walk problems are difficult to solve. See Fig. 1a for an illustration of the random walk of weights.

Short of solving the master equation Eq. 5, it is still possible to gain qualitative insight into some properties of the walk. In particular, we will be interested in the time required for the weight to fall below a certain (small) value w_0 when starting from a different value $w_s > w_0$; this quantity is usually referred to as the *mean first passage time*.

To gain some insight into this, we first review the well known results for the standard discrete time/space homogeneous random walk with bias p (q) to take a step to the right (left). The domain of the random walker is infinite, ranging from $-\infty$ to $+\infty$. The master equation for this random walk is:

$$P(x,t+1) = qP(x+1,t) + pP(x-1,t) \tag{6}$$

We used here x as the state variable, so as to make clear that this is not a model of the neuron. A standard result for this type of random walk concerns the average waiting time of the random walker to reach the site immediately to the left of its starting place. With probability q, this happens immediately at the first step, but the walker could also make an excursion of arbitrary length to the right, before returning to its starting place and finally taking the step to the left. The precise mean first passage time depends on the bias p. Qualitatively, there are two regimes: the mean first passage time is small when $q > p$, but diverges when $p \geq q$; see Fig. 1b for an illustration of this.

The random walk of the spiking neuron (Eq. 5) is different from the standard random walk (Eq. 6), in that the step size of the former changes depending on the location. While the random walker can still make an infinite number of steps into each direction, it is restricted to the interval [0, 1]. As it reaches one end, say 0, the step size towards the boundary will become smaller, such that it never reaches the boundary. More precisely, the ratio of the step size to the right (increasing the weight) and to the left (decreasing the weight) is given by $(1-w)/w$. From this it can be seen, that the step sizes are equal only when the weight is $w = 0.5$.

In order to relate this inhomogeneous random walk to the standard one, we now ask how many steps to the left are required, so that the weight is reduced by $(1-w)\epsilon/(1+\epsilon)$ as a single step in the positive direction. To simplify the reasoning, we can assume that this step would leave the weight within the allowed range. An estimate of the required number of steps is

$$\#\text{steps required} = \frac{1-w}{w} \tag{7}$$

In reality, the number will be higher, because as the neuron reduces the weights, the weight decrements become smaller as well, but this simplification does not alter the conclusion we will reach. If the distance covered by a step in the positive direction is Δ, and the probability to make a step to the left is q, then the probability to make the required number of steps to travel by Δ in the negative direction is $q^{\frac{1-w}{w}}$. On the other hand, the probability that the walker takes a single step into the positive direction during this period is given by $1 - q^{\frac{1-w}{w}}$. Thus, locally, the random walk Eq. 5 in weight space looks like a standard random walk Eq.6 with appropriately modified transition probabilities. Note, that this conclusion remains valid even if the probability q becomes a function of w, for as long as $q(w)$ does not go to 1 as $w \to 0$.

We observe that the local transition probability $q^{\frac{1-w}{w}} \to 0$ as $w \to 0$, and hence we know that there is an area of small

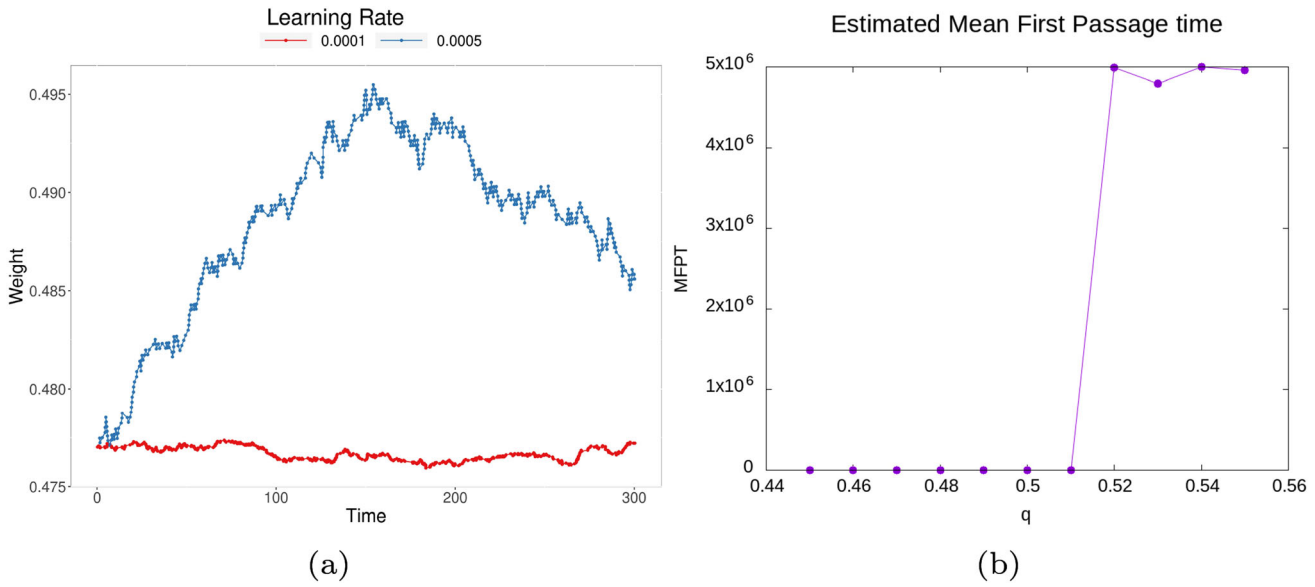

Fig. 1 **a** Illustration of the random walk of an STDP neuron. We simulated a neuron with 2 weights for 300 time units and two different learning rates, $\epsilon = \{0.0001, 0.0005\}$. The y-axis shows the value of the first weight. **b** The mean first passage time of the simple random walk (Eq. 6) to reach the site immediately left to the starting site estimated based on 2000 samples and maximal run times of 5 million steps. The horizontal axis is the probability to take a step to the right. The rapid transition from short to quasi-infinite times is apparent. At $p = 0.5$ transition still seems to be short, which is because the average in this case is dominated by rare walks with infinite excursions

weights to which the mean first passage time diverges. Based on this, we can formulate the following proposition:

Proposition 1 *(First passage time to lower boundary) Let w_s be the weight associated with channel i at time $t = t_0$ and let the weights of the neuron evolve according to the master equation 5 with a learning rate of ϵ. Then, there is a set of values $S := [w_0; w_s]$ with $w_0 < w_s$ such that the first passage time to reach a value $w \in S$ is finite and indeed short. The first passage time for values $w < w_0$ diverges.*

Note that it may be the case that $w_0 = 0$, in which case the weights can reach any of the allowed values within a short period of time. This will be the case, when the learning rate ϵ is sufficiently high. By symmetry, we can immediately formulate the corresponding proposition for reaching an upper boundary.

Proposition 2 *(First passage time to upper boundary) Let w_s be as in Proposition 1. Then, there is a set of values $S := [w_s; w_0]$ with $w_0 > w_s$ such that the first passage time to reach a value $w \in S$ is finite and indeed short. The first passage time for values $w > w_0$ diverges.*

This first passage time property has tangible consequences for the behaviour of the Hebbian neuron, because the Markov chain that the random walker is sampling is not ergodic. It has absorbing states and also absorbing sub-sets corresponding to specific weights being set to zero. We first define what we mean by an absorbing sub-set.

Definition 1 (Absorbing sub-set) Let $\mathbb{V}$ be the space of all possible values of the weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_N)$ of a neuron, with w_i the weight of the i-th input channel of the neuron and $\sum_i w_i = 1$. Let $\mathcal{T}$ be a learning rule, and $\mathbf{w}(t) \in \mathbb{V}$ a weight vector that has been updated t times according to $\mathcal{T}$. A set $\mathbb{A} \subseteq \mathbb{V}$ is absorbing with respect to $\mathcal{T}$ if $w(t_0) \in \mathbb{A} \Rightarrow w(t_1) \in \mathbb{A}$ for all $t_1 > t_0$.

In essence, an absorbing sub-set is a part of weight space from which there is no more escape under the update rules of a training algorithm $\mathcal{T}$. The nature of the absorbing sub-set will depend on the learning rule; indeed, the absorbing sub-set may be the empty set. For our purpose, we find that the Hebbian learning induces a number of absorbing sets on the weight space.

Proposition 3 *(Absorbing sets of the Hebbian neuron) Let $\mathbb{V}$ be the space of all possible values of the weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_N)$ of a Hebbian neuron, and $\mathbb{V}^n$ is a sub-set of $\mathbb{V}$ such that for each weight vector $\mathbf{w} \in \mathbb{V}^k$ there exist exactly k indices j with $w_j = 0$. $\mathbb{V}^k$ is an absorbing set for all $k \leq N$.*

It is straightforward to see this. According to the Hebbian update rule, a weight can only increase if it triggered an output spike. If the weight is vanishing, it cannot trigger an output spike and hence will never be increased.

The absorbing sub-sets $\mathbb{V}^k$ are organised hierarchically in that for every $\mathbb{V}^k$ and $k < N$ there is a sub-set $\mathbb{V}^{k+1}$ such

that $\mathbb{V}^k \subseteq \mathbb{V}^{k+1}$. At the highest level, there are N different absorbing sub-sets corresponding to exactly one of the N weights vanishing. There are a further $\binom{N}{2}$ absorbing sub-sets corresponding to two weights being zero, and more generally $\binom{N}{k}$ sub-sets with k vanishing weights. From this proposition follows the following corollary:

Corollary 1 (*No-escape for the Hebbian neuron*) *Let* $\mathbf{w} \in \mathbb{V}^n$ *with* $n > 0$ *be the weight vector of a Hebbian neuron. If trained under the Hebbian rule, then the weights will be restricted to sub-sets* $\mathbb{V}^m$ *with* $m \geq n$ *for all times.*

This tells us that the Hebbian neuron, once it has set one of its weights to 0, will never be able again to recover from this. From the no-escape Corollary 1 it follows directly that the neuron will, after an infinite number of training updates, end up in a globally absorbing state.

Corollary 2 (*Globally absorbing state*) *In infinite time, the Hebbian neuron neuron will reach a globally absorbing state located in* $\mathbb{V}^N$.

However, note that the passage time to the globally absorbing state may diverge because of Proposition 1.

During training, the Hebbian random walker will often get trapped in metastable states fulfilling $P(i|o) = w_i$ for all weights, and $w_i > 0$ for more than one channel. Each of the metastable states has a basin of attraction. This is a neighbourhood around the meta-stable state with the property that the random walker will be attracted back to the metastable state if it remains within this neighbourhood. We can now formulate the following proposition:

Proposition 4 *(Dwell time in the basin of attraction of a meta-stable state) The random walker will remain in the basin of attraction of a meta-stable state either for a short time or for quasi-infinite times.*

Leaving a basin of attraction is a first passage time problem. In order for the random walker to leave the basin of attraction, at least one of its weights has to fluctuate beyond the boundary of the basin. From propositions 1 and 2 it follows that the time to do this is either very short or quasi-infinite. In practice, this means that the dwell time of the random walker in the basin of attraction of a metastable state will depend on the size of the basin of attraction and the learning rate. The proposition implies that there will be sudden transitions in the transient behaviour as parameters of the neuron are changed. For example, a small adjustment of the learning rate may be sufficient for the dwell time in a meta-stable state to change from quasi-infinite, to very short.

We found that for the Hebbian neuron $\mathbb{V}^k$ are absorbing sets. The following proposition tells us that for the STDP neuron they are not.

Proposition 5 *(STDP metastable states have all weights non-vanishing) Assume an STDP neuron with N input channels receiving statistically independent input spikes at frequencies* $f_i > 0$. *The weight of the i-th input channel is* w_i. *In a metastable state all weights will be non-vanishing, that is* $w_i > 0$ *for all i.*

Assume that the proposition is not true, and that in a meta-stable state channel i has a vanishing weight $w_i = 0$. The assumption of the proposition is that the input channel fires with a frequency $f_i > 0$. By coincidence, it will fire within a time period τ before an output spike. At this point, the weight of the channel will be increased by ϵ. Vice-versa, for as long as $w_i = 0$ the weight cannot be decreased. Hence, the mean step size is non-vanishing and hence cannot correspond to a meta-stable state.

From this proposition it follows immediately that weights of channels cannot remain vanishing for a long time in the STDP neuron.

Corollary 3 (*STDP weights do not vanish permanently*) *Let i be an input channel of an STDP neuron and* $w_i = 0$ *at time* $t = t_0$. *Then there is a time* $t_1 > t_0$ *such that* $w_i > 0$ *at* $t = t_1$.

3.2 Calculating the mutual information

In many practical contexts the mutual information between the input and the output of the neuron is of interest. The mutual information quantifies how much one can learn about the input to the neuron from its output, or vice versa. Depending on the task at hand, one may be interested in maximising the mutual information (Toyoizumiyz et al. 2004), or indeed minimising it (Tishby et al. 1999). Here, we quantify the mutual information between a specific input spike and the output spike (or lack thereof). This notion of mutual information quantifies how much information one has about the last input spike, given that the neuron produced an output spike or failed to do so. Since the output of the neuron can take two values, namely to fire or not to fire, the maximal possible value of the mutual information is 1 bit. There are numerous other ways to define mutual information. For example, one could take into account uncertainty about the time of spikes, or consider the mutual information of sequences of input spikes and output spikes. However, for our current purpose, the mutual information defined in this way is an appropriate measure that will provide useful insights into the dynamics of Hebbian and STDP learning.

More formally, below we calculate the mutual information $I(\mathcal{I}, \mathcal{O})$ between the input $\mathcal{I}$ and the corresponding output $\mathcal{O}$. The input can take an integer value $1, 2, \ldots, N$, where N is the number of input channels. The output takes the values o and $\neg o$ corresponding to an output being

generated following an input or not. $P(o|i)$ denotes the probability that given input channel i fired an output spike will be produced. Similarly, the probability that an input spike through channel i results in the neuron not producing an output spike is $P(\neg o|i)$. Correspondingly, we will be interested in the probability $P(i|o)$ that a particular output was triggered by an input through channel i; similarly $P(\neg o|i)$ is the probability that a non-spike of the neuron follows an input through channel i. We define the mutual information.

Definition 2 (Mutual information between input and output) Let $P(o)$ be the probability that following an input spike the neuron produces an output spike and let $P(i)$ be the probability that input channel i fires. Furthermore, let $P(o|i)$ and $P(i|o)$ be the corresponding conditional probabilities. Using the general definition of the mutual information in terms of entropies and conditional entropies $I(X,Y) = H(X,Y) - H(X|Y) - H(Y|X)$, as well as the general relation $P(i,o) = P(i)P(o|i)$ we then obtain:

$$\begin{aligned} I(\mathcal{I},\mathcal{O}) = &- P(o)\log_2 P(o) - P(\neg o)\log_2(P(\neg o)) \\ &+ \sum_i P(i)P(o|i)\log_2 P(o|i) \\ &+ \sum_i P(i)P(\neg o|i)\log_2 P(\neg o|i) \end{aligned} \tag{8}$$

or equivalently

$$\begin{aligned} I(\mathcal{I},\mathcal{O}) = &- \sum_i P(i)\log_2 P(i) + P(o)\sum_i P(i|o)\log_2 P(i|o) \\ &+ P(\neg o)\sum_i P(i|\neg o)\log_2 Pi|\neg o) \end{aligned} \tag{9}$$

In general, this mutual information is intractactable to compute exactly. For example, it is normally difficult to obtain an expression for $P(o)$. However, some useful general statements can be made about the mutual information.

Proposition 6 *(Mutual information when the spiking threshold $\theta = 0, \infty$) The mutual information vanishes when the spiking threshold is either $\theta = 0$ or $\theta = \infty$.*

To see this let us consider the case of $\theta = 0$ threshold first. In this case, $P(o) = 1 - P(\neg o) \to 1$, which also implies that the conditional spiking probability $P(o|i) \to 1$ for any i, hence Eq. 8 must vanish. By the same reasoning, we can see that the mutual information also vanishes in the limit $\theta \to \infty$. This implies that the mutual information is either 0 everywhere or there is at least one θ which maximises the mutual information. Since the mutual information is not vanishing everywhere, we conclude that there is at least one threshold that maximises the mutual information. This conclusion is independent of the nature of the input data and indeed the training algorithm.

3.2.1 Mutual information in the large threshold regime

In the limit of the neuron having a large firing threshold, it is possible to compute the mutual information analytically.

Definition 3 (Large and small threshold regime) We say that a spiking neuron operates in the large threshold regime when $\theta \gg \sum_i w_i$. Conversely, we say that it operates in the small threshold regime if $1/N \gg \theta \approx 0$.

The next proposition states that in the large threshold regime, the mutual information of a trained Hebbian neuron can be computed based on knowledge of the weights and the input statistics only. In particular, there is no requirement to know the probability of an output spike, which is usually difficult to obtain.

Proposition 7 *(Mutual information in the large threshold regime for a trained Hebbian neuron) Assume a Hebbian neuron parametrised in the large threshold regime. In steady state the mutual information is then given by:*

$$\begin{aligned} I(\mathcal{I},\mathcal{O}) = &- \frac{\sum_i P(i)w_i}{\theta}\log_2\left(\frac{\sum_i P(i)w_i}{\theta}\right) \\ &- \left(1 - \frac{\sum_i P(i)w_i}{\theta}\right)\log_2\left(1 - \frac{\sum_i P(i)w_i}{\theta}\right) \\ &+ \sum_i P(i)w_i \log_2 w_i \\ &+ \sum_i P(i)\left(\sum_{j\neq i} w_j\right)\log_2\left(\sum_{j\neq i} w_j\right) \end{aligned} \tag{10}$$

In order to calculate the mutual information, one normally needs to estimate the probability of an output spike $P(o)$ before calculating the mutual information. In the case of the Hebbian neuron in the large threshold regime, this can be avoided. In this limit the system will have received a large number of input spikes through each of its input channels (on average) between any two output spikes. It is then possible to find an analytic expression for the mutual information. We first calculate the average membrane potential after $L = k \cdot N$ steps, where $k \in \mathbb{N} \gg 1$. The idea is that after L steps each of the input channels has fired a number of times in proportion to its relative firing frequency. We can then write the membrane potential:

$$\langle V \rangle = L \sum_i^N w_i P(i)$$

Here $P(i)$ is the probability that the i-th input channel fires. An output spike happens when $\langle V \rangle = \theta$; we can thus write:

$$\frac{1}{L} = \frac{\sum_i P(i) w_i}{\theta} = P(o) \tag{11}$$

This can be substituted into Eq. 9; remembering that in the case of a Hebbian trained neuron $P(i|o) = w(i)$, we then obtain the proposition.

Mutual information in the small threshold regime It is possible to make some general statements about the mutual information and the weight evolution in the limiting case of very small thresholds θ. In the small threshold regime the numerical details of the weight values do not matter. When an input spike arrives through one of the channels whose weight is above the threshold then the neuron will always spike; otherwise, it will never produce an output. The Hebbian learning rule guarantees that those channels whose weight are below the threshold will never increase their weights any more. The weight normalisation implies that those weights that are below the threshold are driven to zero.

Proposition 8 *Assume a Hebbian neuron parametrised in the small threshold regime. The mutual information can then be computed from knowledge of the weights alone.*

$$I(\mathcal{I}, \mathcal{O}) = -\frac{\sum_i \Theta(w_i - \theta)}{N} \log_2 \left(\frac{\sum_i \Theta(w_i - \theta)}{N} \right) - \\ - \frac{\sum_i (1 - \Theta(w_i - \theta))}{N} \log_2 \left(\frac{\sum_i (1 - \Theta(w_i - \theta))}{N} \right) \tag{12}$$

Here $\Theta(x)$ is the Heaviside function and evaluates to 0 unless $x > 0$, when it evaluates to 1.

This proposition states that in the small threshold regime, the mutual information reduces to the entropy of the output signal of the neuron. To see this, note that in the small threshold regime an input spike through channel i with $w_i > \theta \approx 0$ will *always* trigger an output spike; if $w_i < \theta$, i.e. $w_i \approx 0$, then it will *never* trigger an output spike. Thus $P(o|i) = 1$ when $w_i > \theta$ and $P(o|i) = 0$ otherwise. This means that the last two terms in Eq. 8 vanish. Furthermore, we also have $P(o) = \sum_i P(o|i)/N$. The proposition (Eq. 12) can then be obtained by substituting the special value of $P(o)$ into Eq. 8.

Corollary 4 (*Maximal value of mutual information*) *The mutual information is* 1 *and it is obtained in the small threshold regime when exactly N/2 input channels have weights below the threshold.*

This is a direct consequence of Proposition 8. Then $P(o)$ evaluates to $\frac{N/2}{N} = 1/2$. and the mutual information reaches its maximal value of 1 bit.

Proposition 9 *(Maximal mutual information in small threshold regime) The mutual information can take its maximal value of 1 only in the small threshold regime.*

This can be seen directly from Eq. 8. The first two terms, are the function $-(x \log_2(x) + (1-x) \log_2(1-x))$, which is known to have a maximal value of 1 when $x = 0.5$. The final two terms will make a negative contribution. Hence, the mutual information can only reach its maximal value when these final two terms vanish, which can only be the case when $P(o|i)$ is either 0 or 1. This latter condition is only true in the small threshold regime.

Another perspective on this is that outside of the small threshold regime, it is no longer true that every input spike through the $N - k$ channels with non-vanishing weights triggers an output spike. In this situation, knowledge of an input spike through one of those channels does not provide certainty about an output spike being generated. Indeed, the higher the threshold, the harder it is to predict the effect of a particular input spike, because input spikes will only sometimes trigger an output spike. Therefore, in this regime input spikes carry less information about output spikes.

4 Results

In this section, we will present simulation results to corroborate and extend our theoretical insights. We will first discuss in detail how various parameters impact on the weight evolution of Hebbian learning. Subsequently, we will make the equivalent investigation for the STDP neuron. We conclude this section by discussing how the results generalise to real world data.

We will be interested in how random effects impact on the weight evolution during learning. In order to understand this, we will focus mostly on Poissonian neurons; See Sect. 2.2 for details. When the input to the neuron is unbiased and uncorrelated, then we know that any bias that evolves in the weights is a consequence of fluctuations, rather than any particular bias in the input data. Throughout most of this article, we will therefore focus on iid input data before confirming that the main conclusions hold when the neuron is trained with real data.

4.1 Hebbian learning

We will first focus on the mutual information between input and output of the neuron. This will provide important

insights into the behavioural regimes of the neuron. Subsequently, we will then characterise the weights the neuron take.

4.1.1 Mutual information

In order to understand how the mutual information depends on the learning rate ϵ and the threshold θ, we performed over 447000 separate simulations of the Hebbian neuron. All results reported here have been obtained for a neuron with $N = 40$ input spikes, unless stated otherwise. The results are summarised in Fig. 2a, where each pixel represent the mutual information as obtained from a single simulation.

The main findings that can be seen from the graph are: For a fixed learning rate, the mutual information has a single maximum value. This means that for a fixed learning rate, there is a unique critical threshold θ^* that globally maximises the mutual information. We find that θ^* is a small value. It gets exceedingly small as the learning rate increases. For higher learning rates, the maximal mutual information is not resolved any more in Fig. 2a. For sufficiently small values of the learning rate ϵ, the mutual information reaches not just a global maximum, but it actually reaches the maximally possible value of 1 bit; cf. Corollary 4. When the threshold is increased beyond θ^*, then the mutual information gradually falls to zero, which is compatible with the functional relationship derived for the large threshold limit above; see Eq. 11 and proposition 6. On the other hand, if θ is decreased from θ^*, then there is a dramatic, almost instantaneous drop of the mutual information to zero, with the mutual information collapsing from its maximal value to 0 for the tiniest changes of the threshold. Due to the small size of the maximising threshold θ, the drop to 0 is only resolved for small learning rates in Fig. 2a. Fig. 2c shows more detail for two particular learning rates.

The mutual information does not vary continuously as a function of the parameters, but the observed values for the mutual information are distributed around specific numerical values only. This is apparent from the heat-map (Fig. 2c), but more clearly seen in Fig. 2c. Taking into account that the weight values are subject to random noise, we take this as meaning that only certain values of the mutual information are allowed. A perhaps surprising consequence of this discrete nature of the spectrum is that the mutual information sometimes changes abruptly as parameters are varied. In Fig. 2a this is readily visible in that the parameter space is organised into discernible regions, which makes the figure reminiscent of phase diagrams as they are used in statistical physics. Finally, we note that a higher learning rate allows for fewer distinct values of the mutual information than smaller learning rates, despite being subject to much higher noise (via the higher learning rate).

4.1.2 Weights

In order to get a better insight into how the parameters of the neuron impact on learning, we next investigate the weights that a single Hebbian neuron can take. A convenient measure to summarise the weights is the entropy of the weights after the system was allowed to settle. To obtain this entropy, we simulated the Hebbian learning system for 60000 time units. Then we interrupted the simulation and recorded the last weights that the system took. Calling these w_i, we calculated the *weight entropy* as $\mathcal{E}_w := -\sum_{i=1}^{N} w_i \log_2(w_i)$. While formally an entropy, this measure does not have a direct information theoretic meaning, but is merely used here as an easy-to-interpret summary statistics of a 40 dimensional weight vector. The upper bound of the weight entropy is limited by the number of weights (which is 40 in this case); the maximal value of $\mathcal{E}_w$ is obtained in the case of maximal disorder, i.e. when $w_i = 1/N$ for all i. The lower bound of the entropy is zero, which is reached when the system is in one of its N globally absorbing states. In-between these two boundaries, the entropy is a continuous function of the weights. We find, however, that Hebbian learning preferentially leads to specific weight entropies only. These correspond to the metastable states.

The weight entropies are summarised in Fig. 2b. It shows regions of constant values separated by abrupt changes, similar to the case of the mutual information. We observe that for ultra-low learning rates, the system retains its full weight diversity (see the bottom blue line in Fig. 2b). In this regime, the random walker is not able to follow a persistent trend and essentially remains stuck around the initial conditions. At the opposite end of the spectrum, when learning rates and/or thresholds are high, the neuron enters into a globally absorbing state which implies that the weight entropy falls to 0. In-between, the system settles onto metastable points. These metastable points are located in absorbing sub-sets of $\mathbb{V}$, that is the neuron has reduced its dimensionality. The learning rate and the threshold dictate which one of the possible points the system takes, in particular which one of the absorbing sub-sets it occupies.

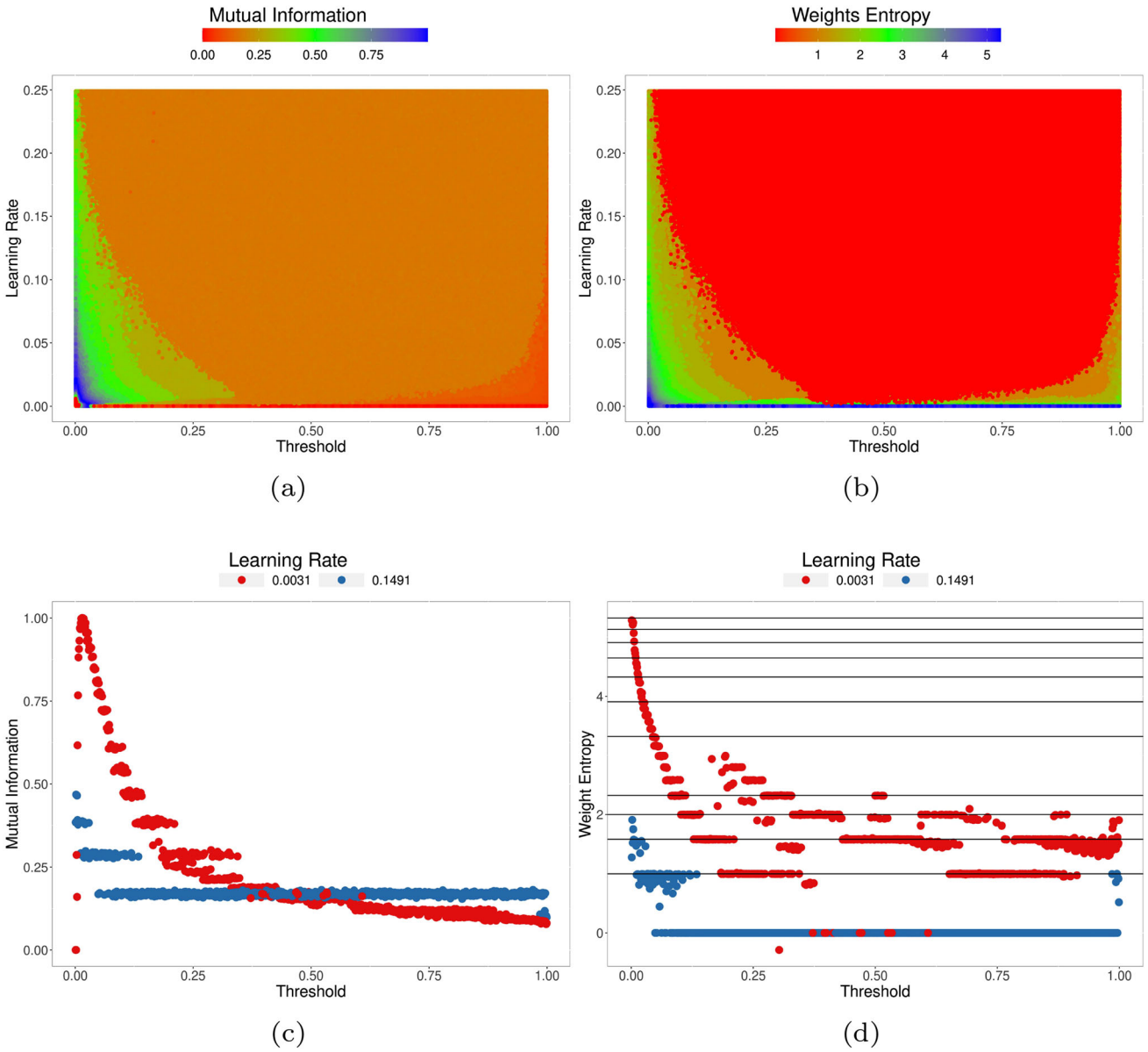

Fig. 2 **a** The mutual information as a function of both the learning rate and the threshold. As the threshold it is varied, it is apparent how the mutual information changes abruptly. **b** The weight entropy as a function of the learning rate and the threshold; the data displayed here represents the same simulations as in Fig. a. The deep red area shows where the neuron reaches the globally absorbing state. **c** Same data, but showing only slices corresponding to $\epsilon = 0.0031$ and $\epsilon = 0.1491$. Here it is clearly visible how the mutual information collapses to 0 as θ is decreased from its maximum. In the case of the higher learning rate, the maximum is no longer resolved. Also visible are the discrete levels of mutual information. **d** Detail of the weight entropy for two different learning rates. The black solid lines indicate the entropy of a system where each weight takes the value $w_i = 1/(N-k)$, and k weights have a vanishing weights. We show lines of $N-k$ equalling 40 for the top line and then $35, 20, \ldots, 10, 5, 3, 2$

4.1.3 Random walk model to explain choice of absorbing sub-set

This begs the question as to what precisely determines how deep the random-walker enters into absorbing sub-sets. In this respect, an important hint is given by Proposition 4, which states that the escape time from a meta-stable state is either fast or quasi-infinite.

We hypothesise that the reduction of the dimensionality is driven by the first passage time of weights to below the threshold θ. At least in the small threshold regime, once the weights w_i of a channel i has fallen to below the threshold θ, this channel i will not be able to trigger an output spike and consequently cannot be increased again. In combination with Proposition 2 this suggests a mechanism by which the weights of channels that are close to the

threshold after initialisation fall below the threshold and never recover back. The idea is that in accordance with Proposition 2, given a learning rate ϵ, some of the weights will be close enough to the threshold θ to have a short first passage time to θ. As more of the weight components w_i fall below zero, the remaining ones get further away from the threshold, putting them into an area where the first passage time to the threshold diverges, in an effect similar to the one depicted in Fig. 1b. In order to test whether the first passage time indeed determines the order k of the sub-set $\mathbb{V}^k$, we compared a random walk model that directly implements the above hypothesis with the Hebbian learning algorithm.

The model is as follows: We assume a random walker $\mathbf{w}$ in weight space with all weights initially drawn from a uniform distribution, before being normalised. After normalisation, we assume that the weights are $w_i > \theta$ for all i. Each weight performs a random walk in weight space. We are interested in the time required for a weight, starting at some value w_0, to fall below a threshold θ when weights are adjusted according to the Hebbian dynamics. We model the random walk of each weight w_i independently. The following routine returns the first passage time:

- Initialise the random walker with a value $w \leftarrow w_0$ and $t \leftarrow 0$; choose a learning rate ϵ.
- Do the following until $t > 2000000$:
  - With probability $1/N$ increase the current value w to $w \leftarrow \frac{w+\epsilon}{1+\epsilon}$. With probability $(N-1)/N$ decrease the weight from the current value w to $w \leftarrow \frac{w}{1+\epsilon}$.
  - $t \leftarrow t+1$
  - If $w < \theta$ stop further iterations and return the value of t.
- Return the value of t.

Here, we limited the duration of the random walk to 2 million iterations, to prevent extremely long running times. This particular value was chosen to be large, but otherwise its precise value is unimportant. In practice we find that the loop runs for the full 2 million iterations or for much shorter, depending on the parameters, as predicted by Proposition 2.

Using this random walk model, we can now determine how many weights of a neuron will fluctuate below the threshold θ within a short time. We can do this using the following model:

1. Draw a set of N random numbers $r_1, r_2, \ldots, r_N$, sort them such that $r_i < r_{i+1}$ and set $r_i \leftarrow r_i / \sum_j r_j$. Set $l \leftarrow 1$.
2. Choose r_l as the initial value for the above random walk model. Determine the first passage time to below the threshold value θ. If this time is ≥ 2000000 then return the value of l.
3. Otherwise, remove r_l from the set, set $l \leftarrow l+1$ and $r_i \leftarrow r_i / \sum_{j=l}^{N} r_j$
4. Go to 2.

The idea of this model is that we start from N input weights above the threshold. Then we check whether one of those (the one with the smallest weight), fluctuates below the threshold θ within finite time (or rather within 2 million time steps). If it does, we check the second value, then the third, etc ...until the first passage time diverges. At this time, we have then determined how many weights remain finite for a long time. Note that the outcome of the model is insensitive to the time limit of 2 million steps, which can be doubled (or halved) without changing the results.

We compared the model predictions (diamonds in Fig. 4) with the simulations of the actual neuron (solid points in Fig. 4). As can be seen from the figure, we found the model to reproduce the real neuron accurately. This corroborates the hypothesis that the absorbing sub-set of the neuron is determined by the first passage time properties of the weights, in the Hebbian neuron.

4.1.4 Multi-stability

So far we have established that the Hebbian neuron transiently (but for very long time) remains in absorbing sub-sets $\mathbb{V}^k$ and that k is a result of the first passage time properties of the random walk. We will now look in more detail into the dynamics and weight distribution of the meta-stable states that the neuron takes. This will uncover unexpected dynamical effects.

To better understand what determines the weights of the neuron, we kept the learning rate fixed and investigated how the weight entropy changed as we varied θ. This gives some insight into which weights the algorithm finds. We chose a learning rate of $\epsilon = 0.0031$, as an example that provides good insight. Figure 2d illustrates that for small thresholds the weight entropies take only discrete values corresponding to homogenous sub-set configurations. By this we mean that the weight vectors $w \in \mathbb{V}^k$ where the weights are either $w_i \approx 1/k$ or vanish. We find that the absorbing sub-sets $\mathbb{V}^k$ become deeper, i.e. k becomes larger, as θ increases, which is expected given the discussion in the previous section. The initially monotonous decrease of the weight entropy is, however, interrupted at a point $\theta \approx 0.125$ where suddenly higher order sub-sets appear again. Note that these "new" weight entropies are again homogenous solutions of higher absorbing sub-sets. However, now there is a bi-stable behaviour, where the neuron

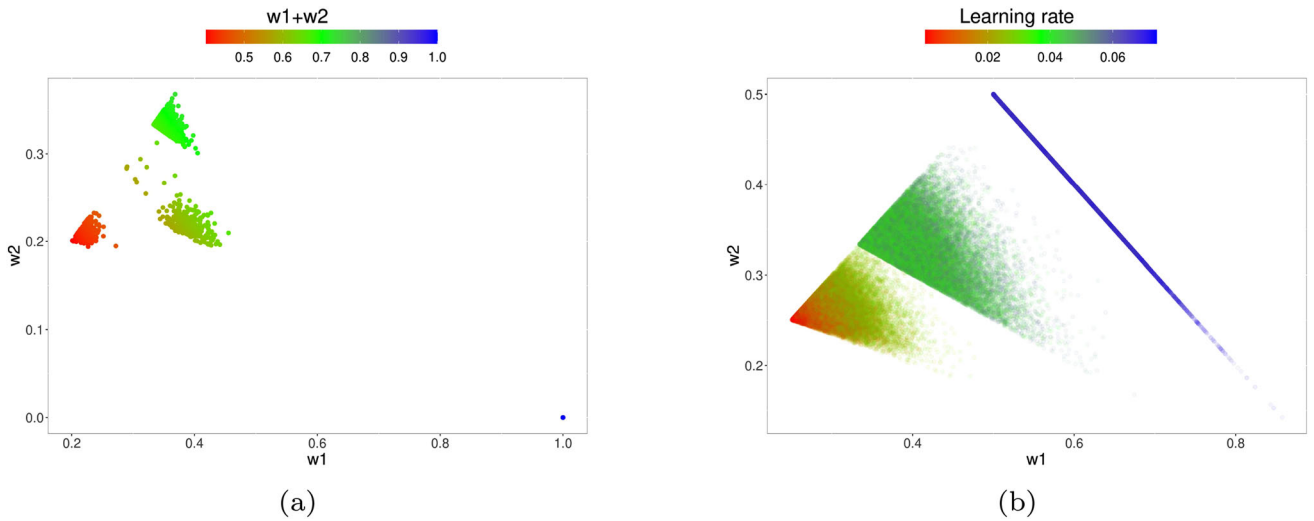

Fig. 3 **a** For $\theta = 0.5$ and $\epsilon = 0.0031$, we ran 2000 simulations for 60000 time units and recorded the weights at the end. Then we plotted the largest weight against the second largest weight, resulting in 4 different "clusters" of weights. Each point represents a single simulation. This demonstrates that for this particular parameter setting, the system stochastically selects between 1 of 4 different solution classes. **b** We started a simulation at a metastable state for $\epsilon = 0.001$ and $\theta = 0.05$ with a learning rate to $\epsilon = 0.005$. Without interrupting the training, we then increased the learning rate every 10000 time units. The graph shows the largest versus the second largest weight sampled every 100 iterations. As the learning rate increases, the excursions from the metastable state become larger until the next metastable state is reached. The transition to deeper absorbing sub-sets are clearly visible

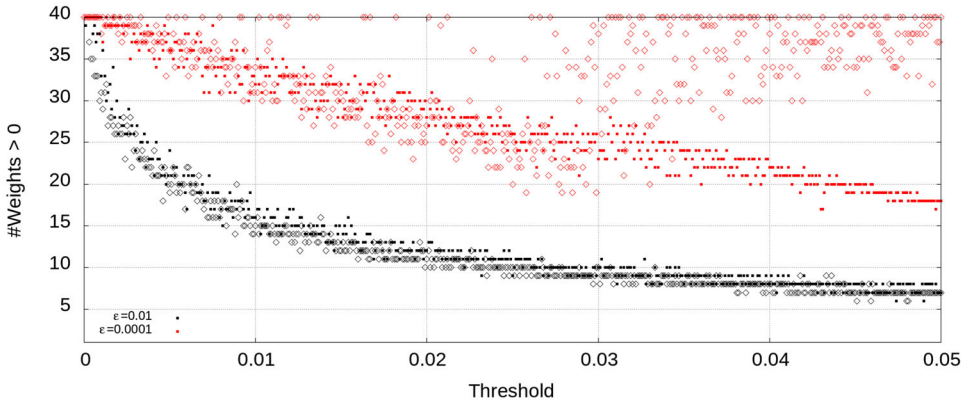

Fig. 4 Comparing the random walk model to the weight evolution of the neuron. The vertical axis shows the number of weights that are expected to be non-vanishing after a long time of training. The solid points are the predicted numbers of non-vanishing neurons from the random walk model for different values of the threshold. The diamonds are the corresponding values from the neuron. We show two different learning rates corresponding to $\epsilon = 0.01$ and $\epsilon = 0.0001$. It can be clearly seen that the random walk model (squares) and the spiking neuron (diamonds) are in excellent agreement

makes a stochastic choice during training whether to enter the deeper or a shallower absorbing sub-set. The figure suggests that two or more different trends operate at the same time. The first trend is the continuation of the monotonous reduction of the dimensionality from the small threshold regime, until the weight entropy reaches zero. The second trend has a similar shape but admits higher weight entropies, corresponding to the neuron settling into higher absorbing sets $\mathbb{V}^k$; see Fig. 3b.

A closer investigation reveals that there are areas of multi-stability of the neuron. In the case of $\theta = 0.5$, the neuron can take 4 different weight configurations. To illustrate this in more detail, we ran 2000 simulations of the neuron for 60000 time units and recorded the largest weight against the second largest weight. The data is plotted in Fig. 3a and shows 4 clusters corresponding to $k = 1, 3, 4, 5$. For each run, the system chose stochastically one of 4 possible solutions. No other solutions were stable.

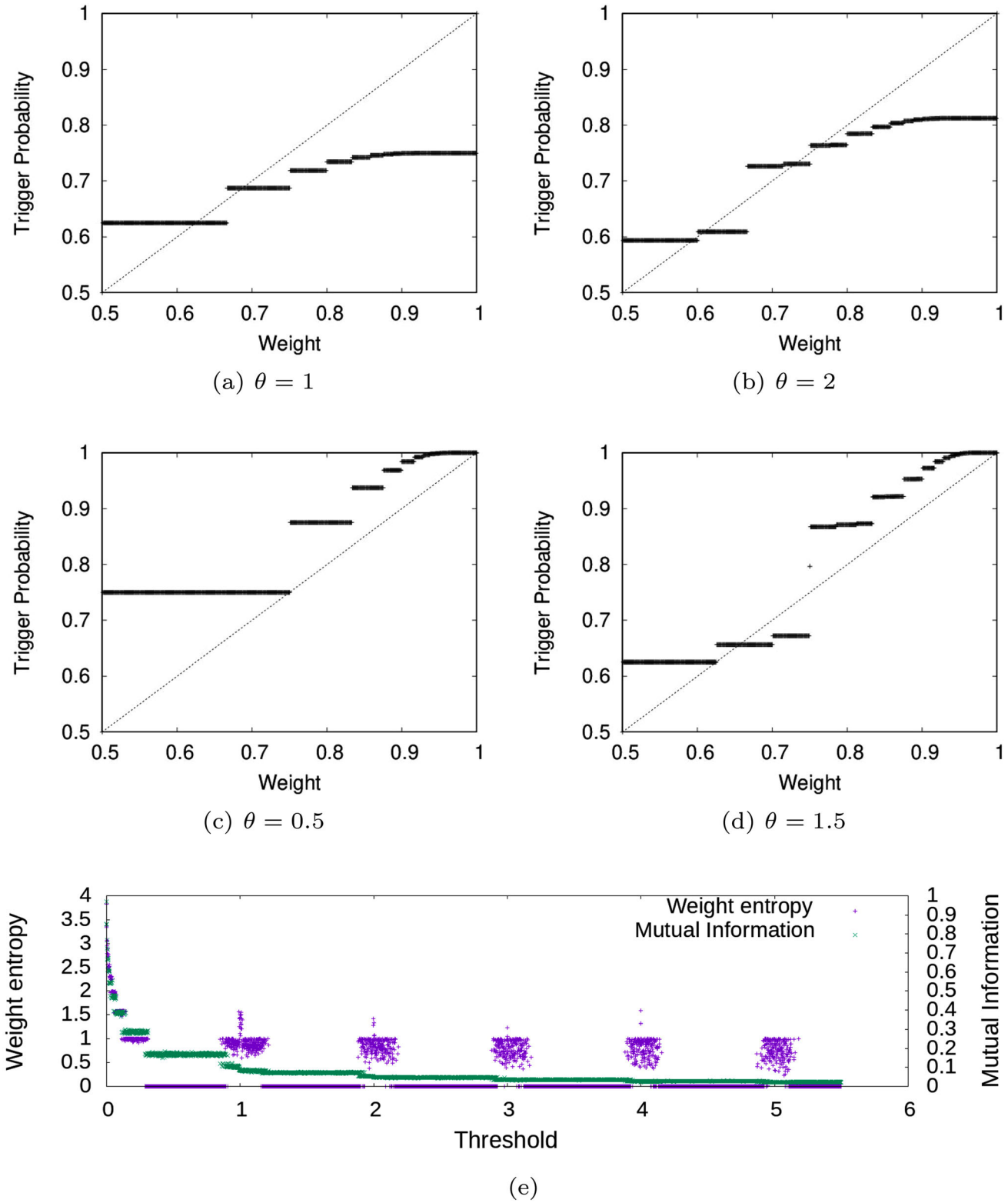

Fig. 5 **a**-**d** Metastable states in the Hebbian neuron: The probability $P(i|o)$ for a channel to trigger an output spike as a function of the weight for different thresholds and a neuron with two weights only. The metastable states correspond to the points where the curve intersects the diagonal. If the probability is higher than the weight, then the weights tend to increase under the Hebbian learning rule. For the case of a threshold of 1 and 2, the curves are mostly under the diagonal, indicating stability, whereas in the case of $\theta = \{0.5, 1.5\}$ the metastable states are not stable. **e** The weight entropy (purple) and mutual information (green) as a function of the threshold and $\epsilon = 0.03$; each point represents a separate training run

4.1.5 Non-monotonicity of the stability of meta-stable states as the threshold is varied

So far, we found a trend that increasing the learning rate and the threshold tends to drive the system into ever deeper absorbing sub-sets, with arguably less interesting dynamics. At least with regards to the threshold, this trend is only true up to a point. As θ is increased sufficiently, the weight entropy reaches zero and the system is in the globally absorbing state. A further increase of θ leads to an *increase* of the weight entropy, albeit only for a narrow interval of thresholds around the value of 1, before it falls to zero again into the absorbing state, then it rises again to around 2 and falls again and so on; see Fig. 5e.

This unexpected behaviour, is a consequence of changes in the stability of meta-stable states, as the threshold is varied. It highlights that the threshold is a key parameter in determining the stability landscape of the Hebbian neuron. To understand this, we investigated the stability of metastable states in the simplified case of a neuron with only 2 input channels. This case is useful to consider because it is tractable in the sense that we can compute $P(i|o)$ explicitly by enumeration. In Fig. 5 we show the exact probability for an input channel to trigger an output spike as a function of the channel weight. Specifically, the figure displays a system consisting of only 2 input channels, each receiving inputs with a frequency of 0.9. This models exactly the situation of a neuron with $N = 40$ input channels out of which $k = 38$ neurons have vanishing weights. In Fig. 5 the metastable states correspond to the weights where the probability to trigger an output spike crosses the diagonal, i.e. where $w_i = P(i|o)$. However, the stability of the metastable state depends on precisely how the curve crosses the diagonal. Taking Fig. 5a as an example, we see that the curve crosses the diagonal four times. The first crossing is at $w = 0.5$, where both weights have equal value; and then there are two more crossings between 0.6 and 0.7; finally, the curve also crosses the diagonal for $w = 1$, which corresponds to the globally absorbing state. The important feature of this figure is that after the third crossing (below 0.7), the curve remains below the diagonal. This means that any perturbation of the weights upwards from the metastable state at around $w \approx 0.7$ is pulled back to this metastable state. In fact, the absorbing state $w = 1$ will only be reached if the weight reaches 1 exactly, otherwise, the weights will converge back to the metastable state. Similarly for the case of $\theta = 2$. A qualitatively different picture emerges for $\theta = \{0.5, 1.5\}$, where the curve is above the diagonal. The two metastable points (one at $w = 0.5$ and one between 0.7 and 0.8) are entirely unstable. Any perturbation of the weights away from the metastable points, will drive the system to the globally absorbing state. A similar explanation can be constructed for other sub-sets. However, calculating the probabilities $P(i|o)$ as a function of the weights and visualising the results becomes challenging for neurons with more than 2 input channels.

We conclude that the periodic change in the weights as the threshold is increased reflects changes in the stability of the meta-stable states. This suggests that, given fixed input data, the stability, and indeed the size of the basin of attraction of meta-stable states is primarily determined by the threshold parameter. Altogether, thus a picture emerges where the absorbing sub-space that the Hebbian neuron enters is determined by the learning rate and the threshold. The latter controls the size of the basins of attraction, whereas the former scales the fluctuations that the weights suffer and as such controls whether the first passage time beyond the basin is short or quasi-infinite.

As an interesting aside, note that while the weight entropy changes non-monotonically, as the threshold increases, the mutual information falls monotonically to zero; see Fig. 5e. Just from looking at the mutual information, one would not be able to infer the changes in the weights as the threshold changes.

4.2 STDP

Turning now our attention to the STDP algorithm, we find three key differences in the behaviour.

- In the case of Hebbian learning, we found that weights space is partitioned into qualitatively different phases. Each phase is a well delimited area of the parameter space where the dynamics is insensitive to changes of the parameters. This is no longer so for the STDP neuron, which takes continuous values for the mutual information.
- The sub-sets $\mathbb{V}^k$ are not absorbing for $k < N$. The STDP algorithm still has a tendency to set a number of its weights to zero. Channels that have vanishing weights eventually recover again, taking non-vanishing values.
- For most parameter settings the STDP algorithm does not settle onto meta-stable states.

It is immediately apparent from Fig. 6a that the mutual information varies continuously as the threshold and the learning rate are adjusted, which is also confirmed by a detailed investigation of the data (not shown). This marks a key difference to the Hebbian neuron. At the same time, Fig. 6b shows that for higher learning rates and thresholds STDP tends to set a large number of its weights to zero, which prima facia is similar to the Hebbian neuron. When considering the update rules of STDP, it becomes clear that the vanishing weights are, and have to be, of a different nature than in the Hebbian neuron. Corollary 3 states that

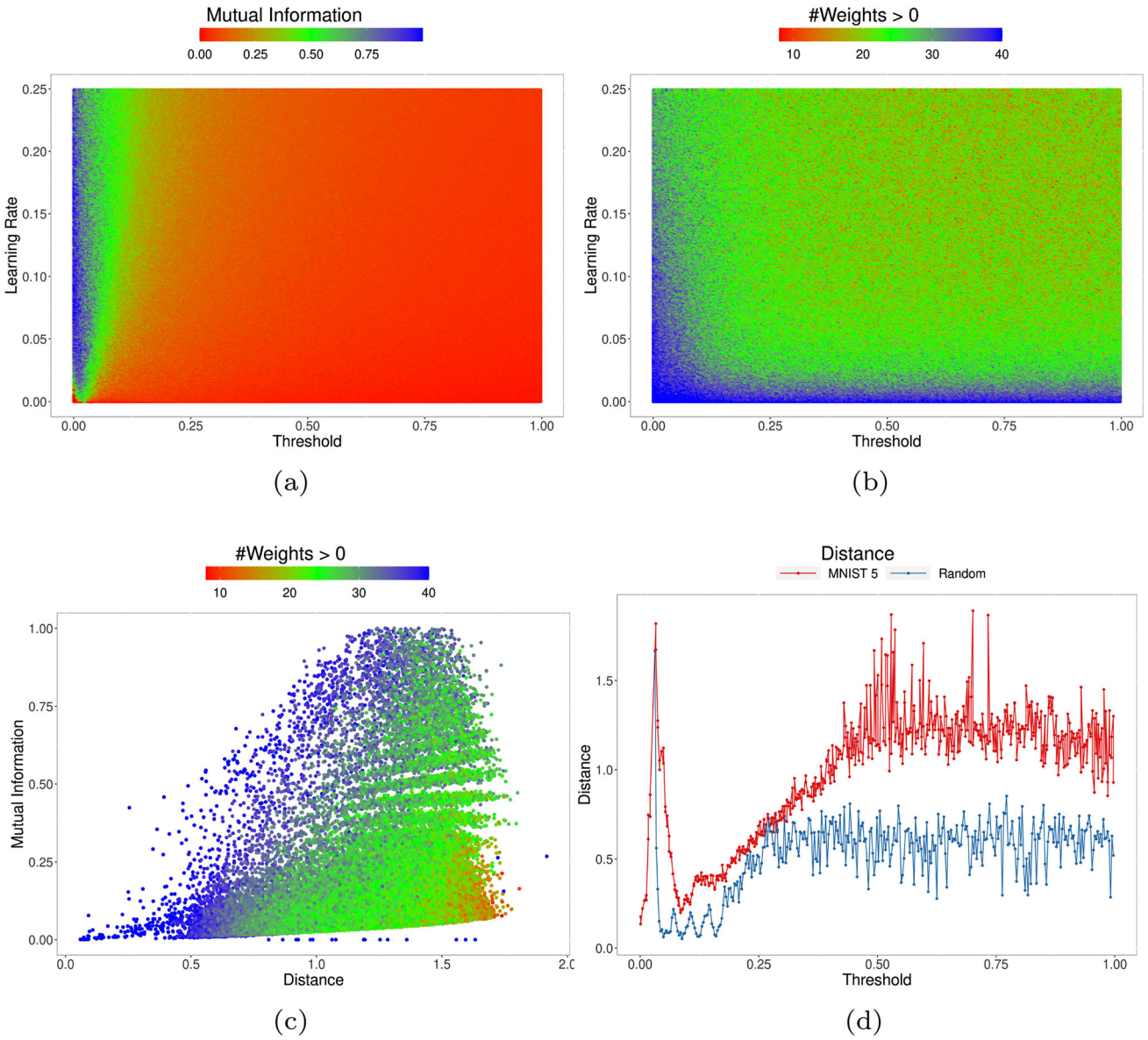

Fig. 6 **a** The mutual information for the STDP neuron as a function of the threshold and the learning rate. Here we show data for a neuron with 40 input channels, each firing at a frequency of 0.9; the window size of the STDP update is 0.1. **b** For the same simulation, we show the number of weights that are non-vanishing. Clearly, there is a continuous trend for the algorithm to set weights to 0, as the learning rate and the threshold (or both) increase. Meta-stable states can only exist where all weights are non-vanishing. **c** The mutual information as a function of the distance. The neuron only remains close to the metastable state when the mutual information is almost zero and when all weights are non-vanishing. **d** The distance from a metastable state as a function of the threshold for the STDP neuron trained on random input data and MNIST (digit 5). The learning rate was set to 0.0001. There is threshold that minimises the distance. Qualitatively, the random data and the MNIST data are the same, but there are quantitative differences

the weights of the STDP neuron cannot remain vanishing permanently. Intuitively, it can also be seen that this must be so. In the simulations we present here, the window length was set to $\tau = 0.1$. The input data is iid, firing with a rate of 0.9 per channels. There will therefore be on average ≈ 2.5 input spikes within $\Delta\tau$ time units before each output spike. This means that each weight is increased about every 11 output spikes by chance alone. To corroborate Corollary 3, we performed additional simulations where we monitored weights over time (data not shown); these simulations confirmed that weights that vanished at one point in the simulation, recovered again to significant values.

The finding that over large parts of parameter space input channels have vanishing weights combined with Proposition 5 means that in those areas the STDP neuron cannot be in a metastable state. To understand this better we checked in simulations the distance of STDP trained neurons (Eq. 4) after a training period of 60000 time units;

see Fig. 6c. A distance of 0 indicates that the weights are exactly in a metastable state. Due to stochastic perturbations of the weights, the neuron will never have a distance of 0, but can come very close to it. Our simulations indicated that across most parts of parameter space the STDP neuron has high distances, which means that it does not settle onto metastable states.

A closer analysis shows that there is an optimal threshold θ that minimises the distance; see Fig. 6d. Furthermore, we found that there is only one, relatively small area of the parameter space where the system evolves towards and remains close to a metastable state. Interestingly, this optimal area is characterised by a vanishing mutual information; see Fig. 6c.

We next discuss the mutual information. By the same reasoning as in the Hebbian case, we would expect from Eq. 8 that the mutual information of the neuron is zero for $\theta \to 0$ and $\theta \to \infty$. Again, this means that there is at least one maximum of the mutual information for intermediate thresholds. In Fig. 6a this is clearly visible as the blue strip on the left hand side of the graph. A closer analysis of the data shows that, similar to the case of the Hebbian neuron, the mutual information reaches the maximal value of 1. It is instructive to analyse how this maximal mutual information is reached, as there are some differences between the case of the Hebbian neuron and that of the STDP neuron: Firstly, we find that for a given set of parameters, the neuron does not reliably reach the maximal mutual information, but it does so only for some repetitions. Secondly, when it does, then this does not necessarily mean that half of the weights are vanishing. Instead, we find that the mutual information is maximal when the probability $P(o|i)$ that an output is generated following an input at channel i is either 0 or 1.

For very low thresholds the mutual information drops to zero, as required by Proposition 6. The critical values where this drop happens become exceedingly small as the learning rate increases. For example, for a learning rate of $\epsilon = 0.05$ and a threshold of $\theta = 10^{-8}$ we found the mutual information to be ≈ 0.2 falling to zero only at $\theta = 10^{-9}$. When we double the learning rate to $\epsilon = 0.1$ then at $\theta = 10^{-13}$ the mutual information is still larger than 0.3 dropping to zero only at $\theta = 10^{-14}$. The drop of the mutual is not resolved in Fig. 6a, except for the smallest learning rates. As the learning rate increases it takes ever smaller values of the threshold to reach a vanishing mutual information. Still, setting the threshold strictly to zero always reproduces the predicted result of a vanishing mutual information.

4.2.1 Real world data—MNIST

So far we have discussed experiments, where all channels receive identical and uncorrelated input. Real-world data is statistically different from random data. It is therefore reasonable to wonder whether the conclusions that we reached from the random data will carry over to realistic input data.

We first consider the case of Hebbian learning. The important qualitative feature that we found above was that the meta-stable state that was chosen by the algorithm depended on the learning rate. This is a consequence of the mathematical properties of the underlying inhomogeneous random walk, or more specifically of the first passage time properties. We would therefore reasonably expect to observe the same behaviour in the case of real data, except that the distribution of meta-stable states may be different then. Real data sets will be less random in the sense that they will have correlations between input channels and over time, which will have an impact on the distribution of meta-stable states. To see why, consider the (trivial) case of a spiking neuron where only channel 1 receives input. This example is minimally stochastic. Clearly, such a neuron will not have any meta-stable states. Independently of the learning rate the weights will converge to $w_1 = 1$ and all other weights to 0. Real world data will be somewhere in-between this minimally stochastic data source and the iid random data that we used above. We therefore conjecture that it will have some meta-stable states, but fewer than the iid random data.

To test this conjecture on an example, we trained the Hebbian neuron on the well known MNIST data-set (Lecun et al. 1998); see Method section for details on how the experiments were set up. We chose this dataset because it is well-known, easily available, reasonably large and commonly used as a benchmark.

As can be seen from Fig. 7, the behaviour of the mutual information is qualitatively the same as in the case of Poissonian neuron. A comparison of Fig. 7a with Fig. 2a, however shows that the behaviour is noisier. The main feature we found for the Poisson data, that the spectrum of the mutual information is discrete, remains valid also in the case of MNIST; see Fig. 7b.

In the case of the STDP neuron we found that the weight evolution did not get captured by meta-stable states when trained on completely random data. One could take the perspective that this is not surprising, because the completely random data did not have anything to learn, so it is reasonable to expect that the STDP algorithm did not learn anything. However, when trained on the MNIST data, we observed the same. The STDP neuron did not converge to any meta-stable states for most settings of the learning rate and the threshold. Indeed, the distance, which is an

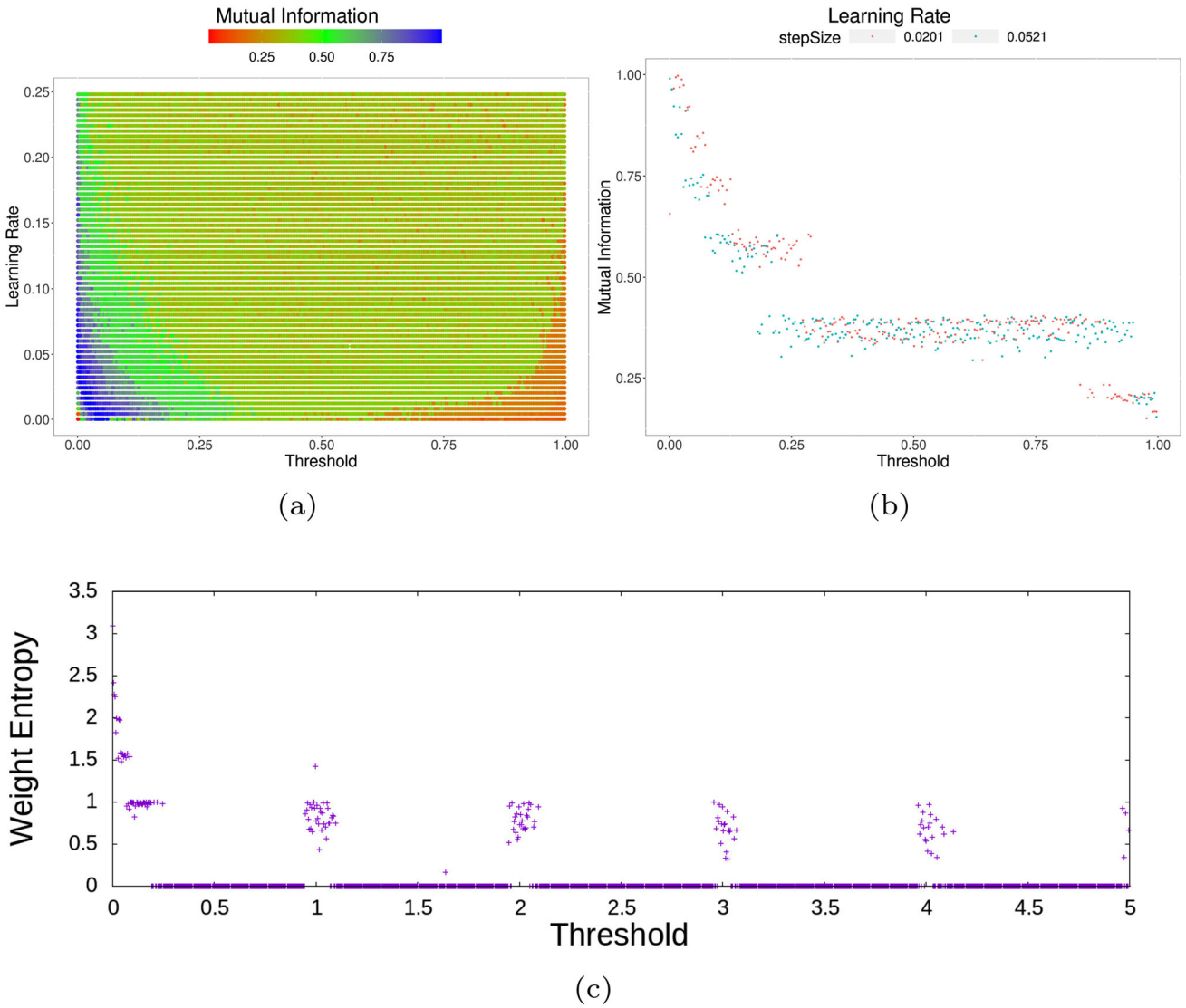

Fig. 7 **a** As in fig. 2b but instead of random input data, we used images of the digit "5" from the MNIST dataset. **b** Same data but only two different learning rates are shown. It is clear that even in the case of the MNIST data, the mutual information is discrete. **c** The entropy of weights as a function of the threshold for the MNIST data. As in the case of the iid input (Fig. 5e), the stability of solutions, and hence the resulting solutions change as the threshold is increased

indicator of how far the neuron is from a stable state, is systematically higher when the neuron is trained on MNIST when compared to the neuron trained on unstructured data; see Fig. 6d. The failure to find metastable states extends to low learning rates. We find, however, that there is an ideal threshold that minimises the distance. As can be see from Fig. 6d, the minimising threshold is the same in the case of unstructured data and the MNIST input.

We conclude, that the qualitative features that we found for both the Hebbian and the STDP neuron, remain valid, at least some types of real world data.

5 Discussion

We established that the dynamics of Hebbian and STDP learning are dominated by stochastic effects. Their behaviour depends crucially on the learning rate, which controls the amount of fluctuations (or noise) that the random walker is subject to, which in turn determines the qualitative behaviour of the algorithms. This suggest that stochastic effects need to be taken into account when analysing the properties of Hebbian and STDP learning algorithms. In particular, whether or not the neuron enters a metastable state and for how long it remains there depends on the size of fluctuations that it suffers and the size of the basin of attraction. The former is controlled by the learning

rate, whereas the latter is due to the threshold. Hence, together, these two parameters are responsible for the transient behaviour of the learning neuron.

An interesting implication of this is that the learning rate has a far more important role in Hebbian and STDP learning than is traditionally acknowledged. It is not merely controlling the speed and precision with which a metastable state is reached, which would be the usual way to think about learning rates in computational intelligence. Instead, the learning rate is crucial in determining *which* metastable state will be taken by the neuron. In the case of the STDP algorithm, setting the learning rate and the threshold correctly is essential so that the algorithm converges even into the neighbourhood of a meta-stable state. To some extent, this aspect of the learning rate in Hebbian algorithms has been pointed out before. It is, for example, well known that Oja's rule only finds the principal components of data in the limit of a vanishing learning rate, but computes something else otherwise. There are few systematic studies of this, however.

A surprising outcome of our study is that the Hebbian neuron displays distinct "phases" in parameter space with different behaviours and sudden transitions between them. These phases are essentially due to a hierarchy of metastable states with different stabilities. In contrast, the STDP neuron does not have phases and for most parts of the parameter space does not settle onto metastable states. Surprisingly, the distance of the STDP neuron from the metastable state depends on the threshold, in that for every learning rate there is a value of the threshold that minimises the distance. Still, for most learning rates, the STDP neuron remains far from the metastable point even at the minimising threshold.

One may now speculate how these different qualitative behaviours impact on the abilities of the neuron to learn. The mutual information, as formulated here, quantifies to what extent one can predict the output behaviour of the neuron from knowledge of the input. For some parameter settings, this quantity can reach the maximally possible value of 1, which means that from knowledge of a single input spike one can determine with certainty what the output will be. It is unlikely that the neuron can perform interesting computation in this regime. A mutual information of 1 means that the neuron computes each input spike separately, without taking into account temporal or spatial correlations. In the other extreme, when the mutual information goes to zero, then the output behaviour of the neuron cannot be predicted at all from a single input. This is compatible with the neuron taking into account complex correlations of inputs. We therefore conjecture that interesting computation requires a low mutual information, although, low mutual information is not sufficient for interesting computation. In this context it is interesting to note that the STDP neuron, finds its metastable only when the mutual information vanishes, suggesting that this marks a sweet-spot for computation. In practice, the combined measure of mutual information and distance can be used to check how well an STDP algorithm has converged to a useful solution; in other words, whether the unsupervised learning has extracted features from the data. A failure to find small values for both, would indicate that the hyper-parameter settings of the learning algorithm are not set appropriately. We thus see the results as practically useful. We also remark that the idea of a minimal mutual information chimes well with the well-known principle of information bottleneck (Tishby et al. 1999), which requires also that machine learning algorithms filter out as much information as possible about the input data.

In the case of the Hebbian neuron, the sweet-spot is harder to identify. There is no point in phase space that minimises the distance, because the Hebbian neuron finds a metastable state for all parameters. It reaches a vanishing mutual information in 3 different cases. Firstly, in the limit of a vanishing threshold. Secondly, in the limit of a vanishing learning rate. Thirdly, in the limit of an infinite threshold, but finite learning rate. Clearly, the first two cases are trivial, because simply no learning takes place in those two limits. The third case is problematic as well, because in this limit the Hebbian neuron tends to fall into the absorbing state, almost irrespective of the learning rate.

We focussed here mainly on the (Poissonian) spiking neuron with iid input. This allowed us to attribute the outcome of simulations to built-in bias of the neuron, rather than biasses in the input data. However, we found that we could reproduce qualitatively similar outcomes with the MNIST dataset as input. This gives confidence that the behaviours we report here are relevant beyond the special assumption of iid data.

The present paper highlights a number of avenues for future research. Firstly, computing meta-stable states directly is intractable, but it may be possible to develop efficient approximations to do this. Secondly, another challenge for future research is to understand how the impact of noise depends on the particulars of the neuron model. Throughout this paper, we considered weight update schemes with normalisation. It would be important to develop a general mathematical theory for other types, or even arbitrary, types of update schemes. Finally, the scope of this contribution was limited to individual neurons. An obvious extension would be to consider networks of neurons. Clearly, in a network, the input to neurons would be the outputs of other neurons in the network. An interesting consequence of this is that the meta-stable states have to be consistent with one another, a situation that is reminiscent of spin-glasses in statistical physics. The

question is whether such networks will then show similar effects, such as genuine phase transitions.

Declarations

Conflict of interest The author has no competing interests to declare that are relevant to the content of this article.

References

Akil A, Rosenbaum R, Josić K (2021) Balanced networks under spike-time dependent plasticity. PLoS Comput Biol 17(5):e1008958. https://doi.org/10.1371/journal.pcbi.1008958

Akil A, Rosenbaum R, Josic K (2020) Synaptic plasticity in correlated balanced networks. arXiv:2004.12453

Białas M, Mirończuk MM, Mańdziuk J (2020) Biologically plausible learning of text representation with spiking neural networks, 433–447 (Springer International Publishing)

Caporale N, Dan Y (2008) Spike timing-dependent plasticity: a hebbian learning rule. Ann Rev Neurosci 31(1):25–46. https://doi.org/10.1146/annurev.neuro.31.060407.125639, pMID: 18275283

Chu D, Nguyen HL (2021) Constraints on hebbian and STDP learned weights of a spiking neuron. Neural Netw 135:192–200. https://doi.org/10.1016/j.neunet.2020.12.012

Davies M et al (2018) Loihi: a neuromorphic manycore processor with on-chip learning. IEEE Micro 38(1):82–99. https://doi.org/10.1109/MM.2018.112130359

Fang W et al. (2021) Spike-based residual blocks. *CoRR***abs/2102.04159**. https://arxiv.org/abs/2102.04159. 2102.04159

Fil J, Chu D (2020) Minimal spiking neuron for solving multi-label classification tasks. Neural Computation. In press

Gerstner W, Kistler W (2002) Spiking Neuron Models (Cambridge University Press)

Hebb D (1949) The organisation of behaviour (Wiley)

Kempter R, Gerstner W, van Hemmen J (1999) Hebbian learning and spiking neurons. Phys Rev E 59:4498–4514. https://doi.org/10.1103/PhysRevE.59.4498

Kozdon K, Bentley P (2018) The evolution of training parameters for spiking neural networks with Hebbian learning, Vol. ALIFE 2018: The 2018 Conference on Artificial Life of ALIFE 2020: The 2020 Conference on Artificial Life, 276–283

Kreiser R, Moraitis T, Sandamirskaya Y, Indiveri G (2017) On-chip unsupervised learning in winner-take-all networks of spiking neurons. (IEEE)

Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324. https://doi.org/10.1109/5.726791

Leen T, Moody J (1992) *Weight space probability densities in stochastic learning: I. dynamics and equilibria*, 451–458

Lin C et al (2018) Mapping spiking neural networks onto a manycore neuromorphic architecture. SIGPLAN Not 53(4):78–89. https://doi.org/10.1145/3296979.3192371

Lobov S, Mikhaylov A, Shamshin M, Makarov V, Kazantsev V (2020) Spatial properties of STDP in a self-learning spiking neural network enable controlling a mobile robot. Front Neurosci. https://doi.org/10.3389/fnins.2020.00088

Long L (2011) *Adaptive spiking neural networks with hodgkin-huxley neurons and hebbian learning* (IEEE, 2011). https://doi.org/10.1109/ijcnn.2011.6033216

Maass W (1997) Networks of spiking neurons: the third generation of neural network models. Neural Netw 10(9):1659–1671. https://doi.org/10.1016/s0893-6080(97)00011-7

Manette O (2011) Local unsupervised learning rules for a spiking neural network with dendrite. BMC Neurosci **12** (S1). https://doi.org/10.1186/1471-2202-12-s1-p210

Markram H, Gerstner W, Sjostrom P (2012) Spike-timing-dependent plasticity: a comprehensive overview. Front Synaptic Neurosci 4:2. https://doi.org/10.3389/fnsyn.2012.00002

Oja E (1982) Simplified neuron model as a principal component analyzer. J Math Biol 15(3):267–273. https://doi.org/10.1007/bf00275687

Orr G, Leen T (1992) Weight space probability densities in stochastic learning: Ii. transients and basin hopping times, 507–514

Plana L et al. (2011) Spinnaker: design and implementation of a gals multicore system-on-chip. J Emerg Technol Comput Syst **7** (4), 17:1–17:18. https://doi.org/10.1145/2043643.2043647

Rubin R, Monasson R, Sompolinsky H (2010) Theory of spike timing-based neural classifiers. Phys Rev Lett 105(21):1–4. https://doi.org/10.1103/PhysRevLett.105.218102

Shrestha S, Orchard G, Bengio S *et al.* (eds) (2018)*Slayer: Spike layer error reassignment in time*. (eds Bengio, S. *et al.*) *Advances in Neural Information Processing Systems*, Vol. 31 (Curran Associates, Inc). https://proceedings.neurips.cc/paper/2018/file/82f2b308c3b01637c607ce05f52a2fed-Paper.pdf

Tishby N, Pereira F, Bialek W(1999) The information bottleneck method, 386–377 (IEEE)

Toyoizumiyz T, Aiharax K, Gerstner W (2004) Spike-timing dependent plasticity and mutual information maximization for a spiking neuron model