Slow feature analysis with spiking neurons and its application to audio stimuli

Guillaume Bellec, Mathieu Galtier, Romain Brette, Pierre Yger

Journal of Computational Neuroscience manuscript No.
(will be inserted by the editor)

Slow Feature Analysis with spiking neurons and its application to audio stimuli

Guillaume Bellec, Mathieu Galtier, Romain Brette and Pierre Yger

Abstract Extracting invariant features in an unsupervised manner is crucial to perform complex computation such as object recognition, analyzing music or understanding speech. While various algorithms have been proposed to perform such a task, Slow Feature Analysis (SFA) uses time as a means of detecting those invariants, extracting the slowly time-varying components in the input signals. In this work, we address the question of how such an algorithm can be implemented by neurons, and apply it in the context of audio stimuli. We propose a projected gradient implementation of SFA that can be adapted to a Hebbian like learning rule dealing with biologically plausible neuron models. Furthermore, we show that a Spike-Timing Dependent Plasticity learning rule, shaped as a smoothed second derivative, implements SFA for spiking neurons. The theory is supported by numerical simulations, and to illustrate a simple use of SFA, we have applied it to auditory signals. We show that a single SFA neuron can learn to extract the tempo in sound recordings.

Guillaume Bellec
Institut de la Vision, Paris, France INSERM, U968, Paris, France CNRS, UMR7210, Paris, France
E-mail: guillaume.bellec [at] ensta.org

Pierre Yger
Institut d'Etudes de la Cognition, ENS, Paris
Sorbonne Université, UPMC Univ Paris06 UMRS968
Institut de la Vision, Paris, France INSERM, U968, Paris, France CNRS, UMR7210, Paris, France

Romain Brette
Sorbonne Université, UPMC Univ Paris06 UMRS968
Institut de la Vision, Paris, France INSERM, U968, Paris, France CNRS, UMR7210, Paris, France

Mathieu Galtier
European Institute for Theoretical Neuroscience
CNRS UNIC UPR-3293, Paris

Keywords unsupervised learning, plasticity, slow feature analysis

Introduction

A property of sensory cortices is the ability to extract and process invariants in the flow of information they are receiving. The higher the cognitive areas, the more complex the invariants [27, 33]. Finding invariants, or equivalently seeking statistically relevant features in the inputs is an unsupervised learning task performed by the sensory cortices [24]. In the context of object recognition, for example, given several images of a plane and several others of a car, learning to answer the question "is it a car?" when facing a new image is rather complex. To answer correctly, one must have identified an invariant belonging to the "car" category, common in all the images. By taking time into account, one could use the fact that two images appearing one after the other in a natural scene likely originate from the same object [9]. This shows the importance for cortical areas to learn from temporal features, and how time can be used as an extra dimension to extract information.

To address more generally this question of time in unsupervised learning, one can start from the observation that sensory signals tend to vary at a fast time-scale compared to the source of these signals in the environment. For example, the physical location of a sound source varies more slowly than the acoustical pressure. This has led several authors to propose statistical learning algorithms that extract slowly varying features of sensory signals [35, 12, 10, 1], which are also related to psychological theories according to which the basis of perception is the "invariant structure" of the sensory (or sensorimotor) signals [17].

Slow Feature Analysis (SFA) [35, 34] is a simple and yet very successful unsupervised algorithm to capture invariant features from input streams. Literature on SFA provides arguments that the brain might process information in a similar way as SFA. In the primary visual cortex V1, the algorithm has been used to explain the emergence of simple and complex cells, and the observed diversity of their properties [2, 21, 6]. In the somatosensory system, it has also been used to show the emergence of place and head-direction cells [13]. SFA extracts continuous causes of the inputs [32] and such stochastic hypotheses also seem to be compatible with known phenomenon of neural adaptation [30]. In this paper, we will show how those principles can be applied to inputs where time is intrinsically entangled to the signals themselves, i.e in the auditory pathway.

If the brain is able to achieve such representations through unsupervised learning, how is it implemented with spiking neurons? While it has already been shown [29] that Slow Feature Analysis (SFA) can be linked to the trace learning rule [12], and that an implementation of such a trace rule can be theoretically compatible with known experimental facts on synaptic plasticity such as Spike-Timing Dependent Plasticity (STDP) [3, 23, 16], a simulation of rate-based or spiking neurons implementing SFA is still missing. In this work, we show mathematically in a similar way as in [29] and with simulations of rate-based and spiking neurons how an optimization criterion derived from slowness can be implemented by neurons, and we apply this in the context of auditory signals.

While the derivation is similar to [29], the conclusions are slightly different. As in their results, we show that synapses can be governed by a biologically plausible online learning rule capturing the slowest time-varying component of the inputs. Yet substantial extensions of their work are added since we found through theory and comparative simulations that classical asymmetric STDP [3] and trace learning [12] are not sufficient to implement SFA, when sharp post-synaptic potentials are considered, while a plasticity kernel shaped as a smoothed second derivative is.

Material and methods

Notations In all the following $x(t) \in \mathbb{R}^n$ denotes the input signal and s is the output such that $\forall t, s(t) := w^T v(t)$, $w = [w_1, .., w_n]$ being the incoming synaptic weights. To simplify the theory all signals are considered null outside a finite support $[0, T]$. The convolution between signals x and y is written $x * y$, the integration over time is written $\overline{x}$, and cross-correlation is written $\langle x, y \rangle := \int_0^T x(t)y(t)dt$. We write $g_\tau(t)$ for the temporal filter given by $g_\tau(t) := \frac{1}{\tau} H(t) e^{-\frac{t}{\tau}}$, where H is the Heaviside function and τ is a time constant. From the mathematical framework detailed in [14] we interpret convolution between temporal signals as simple matrix multiplication. The transposition notation is extended to the continuous representation of filters by noting $g_\tau^T(t) = g_\tau(-t)$ because the property: $\langle x * g, y \rangle = \langle x, y * g^T \rangle$ will be widely used in our paper. The derivative operator at order (r) is also written as a convolution with $d_{(r)}$ such that $x * d_{(r)} := \frac{d^r x}{dt^r}$. δ_{t_0} is a Dirac distribution centred at time t_0.

Toy example To test the convergence of SFA, we use the toy example defined in [35]. Five inputs $x_i(t)$ are built out of non linear combinations of two sinusoids:

$$\begin{cases} x_1(t) = \sin(2\pi f_0 t) + \alpha \cos(2\pi 11 f_0 t)^2 \\ x_2(t) = \cos(2\pi 11 f_0 t) \\ x_3(t) = x_1(t)^2 \\ x_4(t) = x_1(t)x_2(t) \\ x_5(t) = x_2(t)^2 \end{cases} \quad (1)$$

In all the manuscript, we consider $f_0 = 1Hz$.

The spiking neuron model Each input neuron i generates a Poisson spike train of rate ν_i formalized by a sum of Dirac pulses $\theta_i(t) = \sum_k \delta(t - t_k)$. We can define the empirical rate r_i by counting spikes over a time bin of width τ_{rate}. Formally we use $\phi = H(t)\frac{t}{\tau_{rate}^2} e^{\frac{-t}{\tau_{rate}}}$ ($\overline{\phi} = 1$) as an alpha-function filter such that

$$r_i := \theta_i * \phi$$

Note that r_i is in Hz. For simplicity, all input rates ν_i are scaled to have a time average of r and a standard deviation r_s so that $\nu_i := r + r_s\, x_i$. We assume as in [22, 29, 20] for modelling STDP that the post-synaptic neuron generates a Poisson spike train θ_{out} of rate $\nu_{out}(t)$ that is a linear function of the post synaptic potential (PSP) produced by given spike trains θ_i. In other words, we have a frequency offset ν_0, a constant κ, synaptic weights w_i and a normalized PSP filter ξ ($\overline{\xi} = 1$) that defines the neuron dynamic as

$$\nu_{out}(t) = \nu_0 + \kappa \sum_i w_i \theta_i * \xi \quad (2)$$

The plausibility of this assumption is discussed in [22, 29], and mathematical details are given in the Appendix. For numerical simulations, we chose $r = 100\,\text{Hz}$, $r_s = 80\,\text{Hz}$, $\nu_0 = 100\,\text{Hz}$ and $\kappa = 0.0625$. In Figure 6 $\tau_{rate} = 10\,\text{ms}$ and 2 ms in Figure 9. The PSP filter used in Figures 6 and 9 is a decreasing exponential of time constant 1 ms. Simulations have been performed with the Brian simulator [18], using a time step $\delta t = 0.1\,\text{ms}$.

Time scales assumptions We assume that if δt is the discrete time step of the simulation, τ_{rate} the width of the spike counting filter, τ_{signal} the typical time scale of variation of the useful information hidden in the input, τ_w the time constant of synaptic updates, we have

$$\begin{aligned} \delta t \ll \underset{=\tau_{STDP}}{\tau_{rate}} &\overset{(3,a)}{\ll} \tau_{signal} \overset{(3,b)}{\ll} \tau_w \\ 0.1\,\text{ms} \ll 10\,\text{ms} &\ll 100\,\text{ms} \ll 1\,\text{s} \end{aligned} \tag{3}$$

In the following, τ_{STDP} the width of the plasticity kernel is equal to τ_{rate}. Inequality $(3.b)$ is needed so that an online implementation converges towards the same solution as a batch implementation and $(3.a)$ guarantees that information encoded into the rate does not vary faster than the time required to compute the rate.

Calculation of Ω_{SFA} The exact shape of Ω_{SFA} is calculated starting from the expression

$$\Omega_{SFA} := \phi * d_{(2)} * \phi^T$$

When reversing time $d_{(1)}$ becomes $-d_{(1)}^T$, thus $d_{(2)} = -d_{(1)} * d_{(1)}^T$. And using $\phi := g_{\tau_{rate}} * g_{\tau_{rate}}$, we have

$$\Omega_{SFA} = -g_{\tau_{rate}} * g_{\tau_{rate}} * d_{(1)} * d_{(1)}^T * g_{\tau_{rate}}^T * g_{\tau_{rate}}^T$$

The convolution is commutative so we can group the terms using the smoothed derivative filter $d_{(1,\tau)} := g_\tau * d_{(1)} * g_\tau^T$. This leads to

$$\Omega_{SFA} = -d_{(1,\tau_{rate})} * d_{(1,\tau_{rate})}^T$$

Proof is given in [15] that $d_{(1,\tau)} = \frac{g_\tau^T - g_\tau}{2}$. So when developing the expression of $d_{(1,\tau_{rate})}$ we have

$$\Omega_{SFA} = -\frac{1}{4}(g_{\tau_{rate}}^T - g_{\tau_{rate}}) * (g_{\tau_{rate}} - g_{\tau_{rate}}^T)$$

Developing again, only terms in $g * g$, $g * g^T$ or their transpose remain. When inserting the literal expressions $g_\tau * g_\tau^T = \frac{1}{2\tau} e^{-\frac{|t|}{\tau}}$ and $g_\tau * g_\tau = H(t)\frac{t}{\tau^2} e^{-\frac{t}{\tau}}$, we have

$$\Omega_{SFA}(t) = \frac{1}{\tau_{rate}} e^{-\frac{|t|}{\tau_{rate}}} \left(\frac{|t|}{\tau_{rate}} - 1 \right) \tag{4}$$

Construction of the plasticity kernels In order to test the performance of the plasticity kernel Ω_{SFA} we compare it to control learning rules. To do so, knowing that Ω_{SFA} is defined as $\phi * \Lambda_{SFA} * \phi^T$, with $\Lambda_{SFA} = d_{(2)}$, we change that filter Λ_{SFA} by $\Lambda_{Classic} = d_{(1)}$, $\Lambda_{Hebbian+} = \delta_0$ and $\Lambda_{Hebbian-} = -\delta_0$. Those three filters are implementing a classical STDP like plasticity kernel (asymetric), a purely Hebbian plasticity kernel, and an anti-Hebbian one, respectively. They lead to the three control plasticity kernels

$$\begin{aligned} \Omega_{Classic} &:= \quad d_{(1,\tau)} = \frac{H(t)e^{(-\frac{t}{\tau})} - H(-t)e^{(\frac{t}{\tau})}}{2\tau} \\ \Omega_{Hebbian+} &:= \qquad \sigma_\tau = \frac{e^{-\frac{|t|}{\tau}}}{2\tau} \\ \Omega_{Hebbian-} &:= \quad -\sigma_\tau = -\frac{e^{-\frac{|t|}{\tau}}}{2\tau} \end{aligned}$$

Measure of slowness For each set of parameters and STDP kernel, the batch algorithm is run 20 times on the toy example. Slowness of trial i is measured by the correlation coefficient CC_i between output s_i and the optimal solution $\sin(2\pi f_0 t)$, then the CC_i are squared and averaged geometrically such that we can quantify the similarity to the optimal solution in Figure 8 as $\langle CC \rangle = \Pi_{k=1}^{20} CC_k^{2/20}$. Note that if $\tau_{STDP} = 0\,\text{ms}$, rate is ill defined and we replaced the spike train θ_i by its ideal expected value.

Application to audio processing In Figure 9-11, we used recording sounds x_0 sampled at f_s. Inputs to SFA algorithm are built by constructing delay lines of N channels $x_i(t)$ delaying x_0 by delay $i\tau_{delay}$ such that

$$x_i(t) = x_0(t - i\tau_{delay}) \tag{5}$$

To be more precise, in Figure 9 $f_s = 11\,\text{kHz}$, $N = 256$, $\tau_{delay} = 0.81\,\text{ms}$. To study the influence of parameters, the same simulation is run in Figure 10 with $\tau_{delay} = 8.8\,\text{ms}$. In Figure 11 to capture both pitch and tempo we use $N = 512$ delays with $\tau_{delay} = 1.5\,\text{ms}$.

Results

A number of models have been proposed to explain how unsupervised learning of invariants could be achieved by spiking neurons, either with a normative approach [31, 29, 28] or with a phenomenological one [5, 37, 36]. Slow Feature Analysis (SFA) [35, 34] is a simple and yet very successful unsupervised algorithm to capture invariant features. In Figure 1A, the generic principle

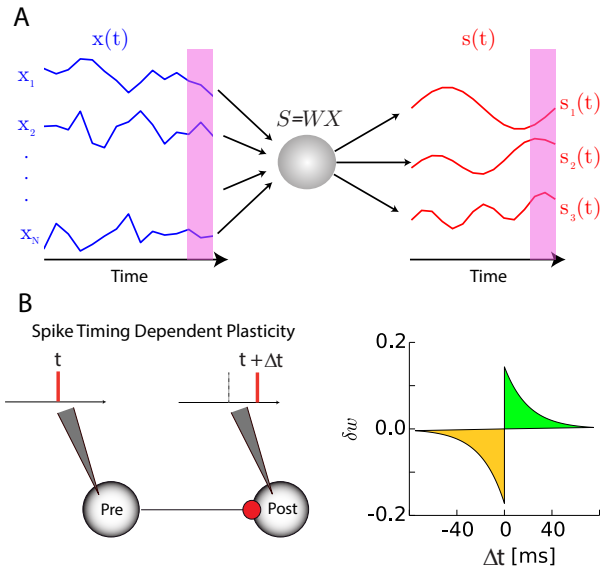

Fig. 1 Illustration of the SFA algorithm and Spike-Timing-Dependent plasticity. **A** If x are the sensory inputs, SFA learns a linear transformation W that maximizes the slowness of the output s. $s(t)$ depends only on $x(t)$ and not on its past. **B** Illustration of the Spike-Timing-Dependent Plasticity protocol, as observed *in vitro* in [3, 23]. The synaptic weight is modified as a function of the temporal difference Δt between pre and post synaptic spikes.

of the SFA algorithm is illustrated. If $x_i(t)$ are the inputs, SFA will decompose it onto a basis of signals $s_i(t)$ sorted by their slowness. In this paper, we focus on how neurons can implement such an algorithm. In line with seminal ideas of Hebbian learning [19], and since its first discovery [4], Spike-Timing dependent plasticity (STDP) appears to be a good candidate to support such a dynamical extraction of invariant by neurons, as a widely observed phenomenon taking place in a large number of systems (see [8] for review). As illustrated in Figure 1B, it relies on the fact that fine pre-post interactions [3, 23, 16] can modify the strength of a single synaptic weight. It has already been stated that some STDP kernel (Figure 1B) can be used to implement SFA [29]. But the question of the most efficient plasticity kernel implementing it remains. Here, we compute analytically what should be the shape of a plasticity kernel implementing SFA and provide simulations of its implementation to compare with kernels suggested in [29].

Implementing SFA with gradient methods

We consider the theoretical formulation of SFA as an optimisation problem. Originally [35] the outputs of SFA are defined as multi-dimensional and decorrelated with each other. However as finding slow components and decorrelating them are two distinct tasks, here we focus only on the first (for the decorrelation task see [26, 7]). Therefore, we assume as in [29] that the inputs x are normalised and decorrelated so that $\langle x, x\rangle = Id_N$. Note that in all simulations inputs are sphered during a pre-processing stage to satisfy this constraint and this is not implemented biologically in this paper (see Discussion). Given those N input signals $x_i(t)$ with zero mean and unit variance, the aim of SFA is to find a set of weights w maximising the slowness of the output $s(t) = w^T x(t)$. This problem is formalised in [35] as

$$\begin{aligned} \min_w \quad & \overline{\frac{ds}{dt}^2} \\ \text{subject to:} \quad & \overline{s^2} = 1 \\ \forall t, \quad & s(t) = w^T x(t) \end{aligned} \tag{6}$$

To illustrate the algorithm, we define a toy example (see Methods), and as can be seen in Figure 2, SFA isolates in the output s the slow component of the inputs. In all the following, this simple example will be used to compare with SFA.

SFA can be re-written so that we can derive a projected gradient algorithm to solve the problem. Because the covariance of the inputs is identity, normalising the weights is equivalent to normalising the output variance. After integrating by part the squared derivative of s, the cost function becomes $-\langle \frac{d^2 s}{dt^2}, s\rangle$[1]. Because $s = w^T x$, and using the notation of the second order derivative operator as a convolution with $d_{(2)}$ (see Methods), we have $\frac{d^2 s}{dt^2} = w^T x * d_{(2)}$. Assuming weights are constant between iterations of a batch algorithm, SFA can be reduced to finding w such that

$$\begin{aligned} \max_w \quad & w^T \langle x * d_{(2)}, x\rangle w \\ \text{subject to:} \quad & ||w|| = 1 \end{aligned} \tag{7}$$

As far as the objective is quadratic in w, the gradient is easily written as $2\langle x * d_{(2)}, x\rangle w$. Beside normalization at each time step, the iteration of the batch algorithm is given by

$$w_k - w_{k-1} = \Delta w_k \propto \langle x * d_{(2)}, s\rangle \tag{8}$$

A key issue of the batch algorithm is that the time flow is not realistic (see Figure 3A). In the batch algorithm, at each iteration the exact same stimulus is presented and the weights are updated based on statistics of the whole sequence. Instead,

[1] The integration by part gives: $\langle \dot{s}, \dot{s}\rangle = [\dot{s}s]_{-\infty}^{\infty} - \langle \ddot{s}, s\rangle$ and the first term is null because the inputs are assumed to be null at infinity.

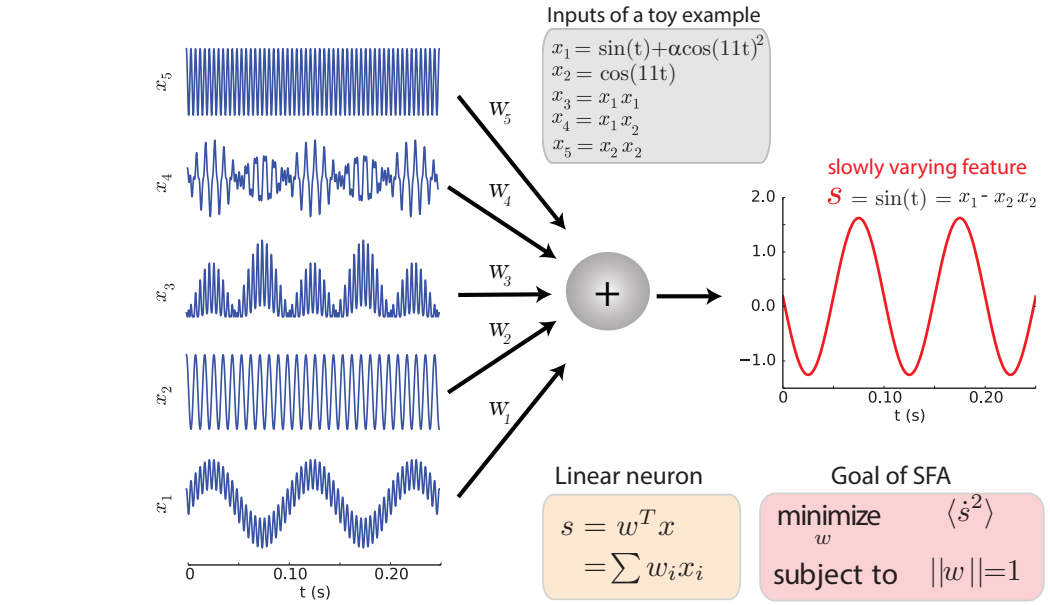

Fig. 2 Illustration of SFA. Simple toy example for the SFA algorithm, as described in [35]. Five inputs (in blue) are non-linear combinations of a slow and a fast sinusoids (see Methods). Output of the SFA algorithm is shown in red. Whitening of the inputs is not shown for clarity, but we have $\langle x, x \rangle = Id_N$

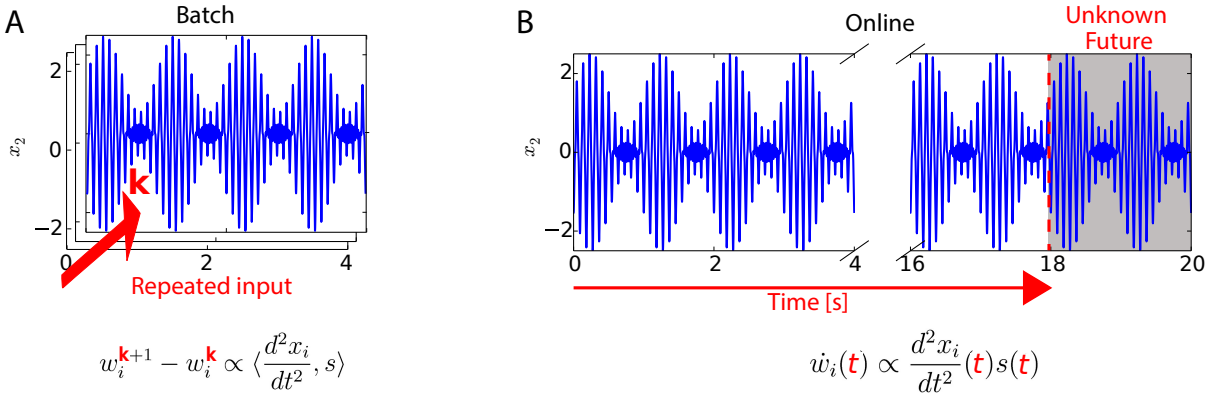

Fig. 3 Conceptual differences between batch and online implementation of SFA. **A** In the batch implementation the same stimulus is presented until convergence, and weights are updated between iterations depending on averaged statistics over the whole sequence. **B** In the online form inputs may evolve over time and not be periodic, weights are updated at every time step depending on the inputs and the output response of the neuron.

as shown in Figure 3B, we can implement an online version of the algorithm, defined by the differential equation of the weight vector where the optimization follows the time flow. If the time constant of weight updates, τ_w, is large enough (as assumed in Equation 3.b), we can define an online algorithm as a Hebbian learning rule of the form $\dot{w}_i(t) \propto \frac{d^2 x_i}{dt^2}(t) s(t)$. One can note that the only difference as compared to Oja's implementation of PCA (Principal Component Analysis) $\dot{w}_i(t) \propto x_i(t) s(t)$ [25] is the use of the second derivative. The same result could have been obtained by keeping one derivative on each side ($\dot{w}_i \propto \dot{x}_i \dot{s}$) but grouping the two make sense when dealing with STDP in the following sections.

In Figure 4, we compare the different implementations of the batch and the online algorithms on the toy example defined in Figure 2. As compared to the ground truth given by the standard SFA algorithm (Figure 4A), the batch (Figure 4B) and the online (Figure 4C) implementations all converge toward the same solution. Moreover, in the case of the online implementation, the dynamics of the weights (Figure 4D) shows that the convergence is achieved in a few seconds of simulated time.

To show that recovering the slow sinusoid in this example is due to the use of the second derivative $d_{(2)}$ in Equation (7), we compared different learning rules on the exact same task (see

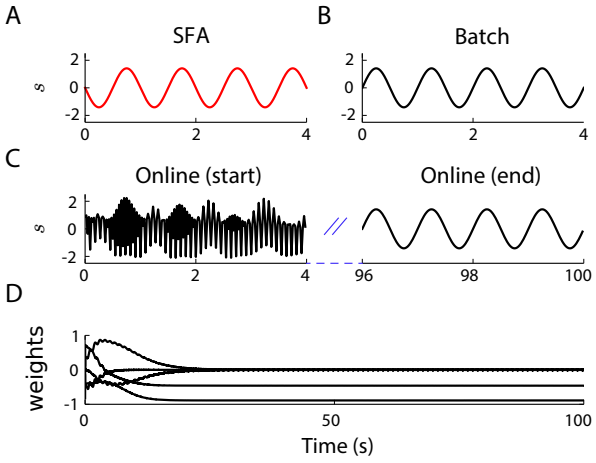

Fig. 4 Convergence of the algorithms on the toy example. **A** Slowest component given by SFA as in [35] for the toy example described in Figure 2. **B** Same but for the batch implementation. (Phases of the oscillations are opposed, but the two solutions are equally optimal). **C** Same for the online form: it converges to the same solution in a few seconds. **D** Evolution of the five corresponding weights as a function of time with the online implementation. Convergence is reached in a few seconds.

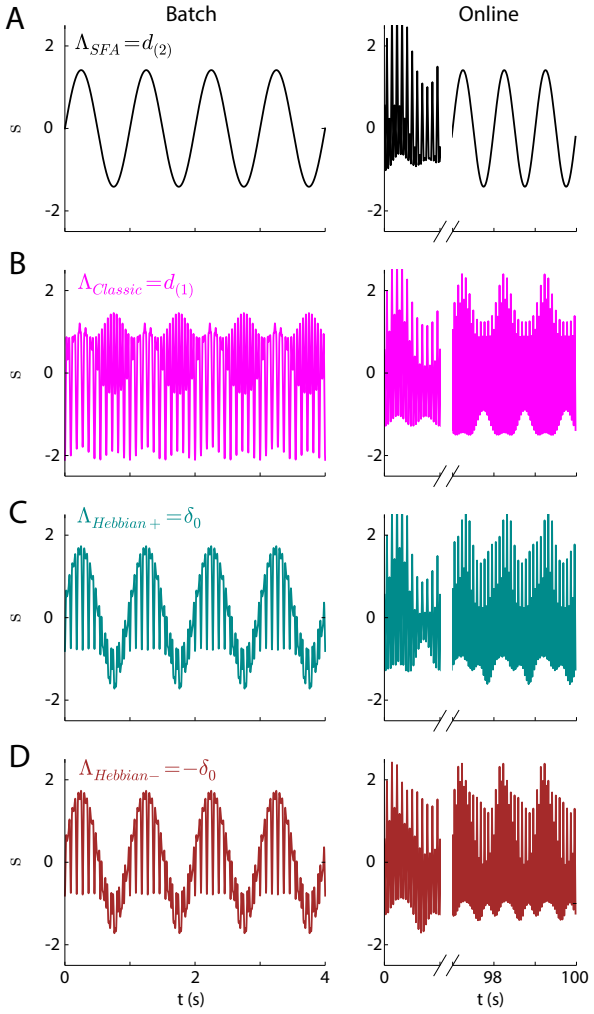

Fig. 5 Comparison with other control learning rules. Four different rules are compared (rows) on the toy example described in Figure 2. Left column shows the output pattern after convergence of the batch algorithm, right shows the results obtained with the online algorithm. **A** SFA kernel as a second derivative. **B** Plasticity kernel based on first order derivative. **C** Hebbian kernel rule implementing PCA [25]. **D** Anti-Hebbian kernel.

Methods) by replacing $d_{(2)}$ in the batch and online algorithm by $\Lambda_{Classic} := d_{(1)}$, $\Lambda_{Hebbian+} := \delta_0$ and $\Lambda_{Hebbian-} := -\delta_0$. Classic stands for the fact that this is at the origin of the canonical kernel for plasticity, as described in Figure 1B [3, 23, 16]. Note that the Hebbian learning rule using $\Lambda_{Hebbian+}$, if normalized properly, is known to recover Oja's neural implementation of PCA [25], defined by $\dot{w}_i \propto x_i * \delta_0(t)s(t) = x_i(t)s(t)$. The left column of Figure 5 shows the pattern of the output after convergence of the batch algorithm, while the right column shows convergence of the online implementation. Only $\Lambda_{SFA} = d_{(2)}$ is able to recover the slow sinusoid.

Implementation by spiking neurons

To propose an implementation of SFA with biological neurons, we need to take into account the spiking property of real neurons. Input x_i is scaled between 20 and 180 Hz to be encoded by pre-synaptic firing rates ν_i. It generates the Poisson spike train θ_i and we can define a rate estimator $r_i := \theta_i * \phi_{rate}$, built by counting spikes over a short time bin (see Methods). We assumed, such as in [22, 29, 20], that the post-synaptic neuron generates a Poisson spike train θ_{out} of rate $\nu_{out}(t)$ linearly related to the post synaptic potential (see Methods). The rate estimator $r_{out}(t)$ is defined as for the pre-synaptic spike trains.

In the following we show that for a kernel shaped as a smoothed second derivative, the expected weight modification due to STDP implements the gradient ascent of Equation (8). Let $t_{i,n}^{in}$ (resp. t_m^{out}) be the spike times of input neuron i (resp. output neuron). If the filter that defines the STDP plasticity kernel is called Ω, and assuming synaptic changes are occurring on a slow timescale τ_w compared to the typical interspike interval, we have, if ϵ is a small learning rate

$$\begin{aligned}\int_{t-\tau_w}^{t} \dot{w}_i(t)dt &= \epsilon \sum_{t^{in},t^{out}\in[t-\tau_w,t]} \Omega\left(t^{out} - t_i^{in}\right) \\ &= \epsilon \int_{t-\tau_w}^{t} \theta_i * \Omega(t')\theta_{out}(t')dt'\end{aligned}$$

Such that, on average

$$\langle \dot{w}_i \rangle = \epsilon \langle \theta_i * \Omega, \theta^{out} \rangle \tag{9}$$

We assume the existence of a kernel Λ so that Ω can be seen as a smoothed version of it, i.e. $\Omega = \phi_{rate} * \Lambda * \phi_{rate}^T$. In particular, the width of such a plasticity kernel Ω is controlled by ϕ_{rate}. If $\Lambda = \delta_0$, then the width of the plasticity kernel is $2\tau_{rate}$. In all the following, to be consistent, we define τ_{STDP} as controlling the width of this STDP kernel ($\tau_{STDP} = \tau_{rate}$). From Equation (9), we have

$$\begin{aligned} \langle \dot{w}_i \rangle &= \epsilon \langle \theta_i * \phi_{rate} * \Lambda * \phi_{rate}^T, \theta_{out} \rangle \\ &= \epsilon \langle \theta_i * \phi_{rate} * \Lambda, \theta_{out} * \phi_{rate} \rangle \\ &= \epsilon \langle r_i * \Lambda, r_{out} \rangle \end{aligned} \tag{10}$$

We see that for $\Lambda = d_{(2)}$, the average effect of our learning rule built with an STDP-like plasticity kernel has a similar effect as the Hebbian learning rule $\dot{w}_i \propto \frac{d^2 x_i}{dt^2} s$ that implements SFA. To generalize the initial batch and online algorithms without spikes [2] we use

$$\Omega_{SFA} = \phi_{rate} * d_{(2)} * \phi_{rate}^T$$

It implies that the plasticity kernel Ω_{SFA} has to be shaped as a smoothed second derivative, and the exact shape of Ω_{SFA} depends only on ϕ_{rate}. The calculation that exhibits the literal expression of the kernel Ω_{SFA} is detailed in the methods, and it leads to the plasticity kernel below

$$\Omega_{SFA}(t) = \frac{1}{2\tau_{STDP}} e^{-\frac{|t|}{\tau_{STDP}}} \left(\frac{|t|}{\tau_{STDP}} - 1 \right) \tag{11}$$

This demonstration gives a good intuition of the reason why this plasticity kernel $\Omega_{SFA}(t)$ implements SFA. However, to see the full mathematical derivation, the reader should refer to the Appendix, where the stochastic aspect of the input and output spiking activities are taken into account, as in [29] using results from [22]. The outcome is that to implement SFA, the expected weight modification due to STDP has to be proportional to $\langle x_i * \Omega_{SFA}, s \rangle$. This is verified by computing the stochastic average of Equation (10). Beside terms that can be neglected (see Appendix or [29]) the expected weight change is in fact proportional to $\langle x_i * \Omega_{SFA} * \xi^T, s \rangle$, where ξ is the post synaptic potential filter. But if the time constant of ξ is small compared to τ_{rate}, the filtering performed by ξ^T can be neglected and STDP using Ω_{SFA} implements SFA. In simulations we used time constants of the order of 1 ms for both ξ and Ω to match biological data and the results are not impaired.

[2] The non-spiking case is recovered if $\tau_{rate} \to 0$ and $\theta_i(t)$ is replaced by the expected value being the rate ν of the inputs.

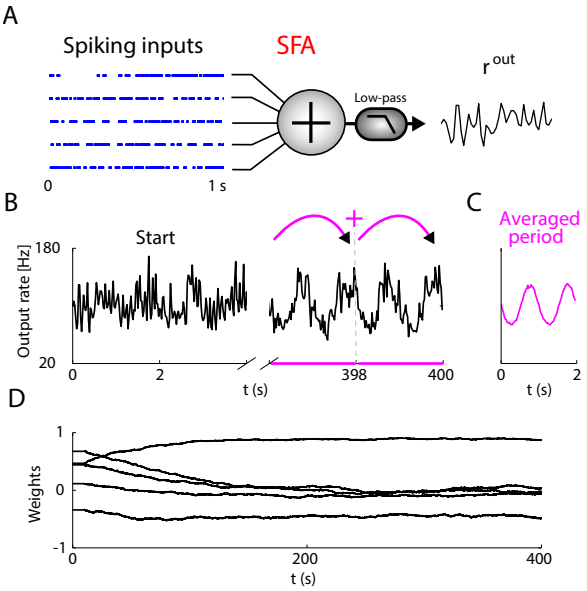

Fig. 6 Generalisation of the online implementation to rate-based neurons. **A** We consider r_{out} as the output of the neuron to represent the slow output. **B** Implementation of SFA for the toy example described in Figure 2 with five inputs being non-homogeneous Poisson processes of rates $x_1, \dots x_5$ re-scaled to vary from 20 to 180 Hz. Evolution of r_{out} during learning. **C** Average over multiple chunks of 2 seconds showing the mean pattern extracted by the algorithm. **D** Evolution of the five synaptic weights during learning, to show that convergence is achieved.

In Figure 6, we test the implementation of SFA with spiking neurons. Figure 6A illustrates the spiking property of the neuron. Variations of the output rate estimator r_{out} over time are represented in Figure 6B. After convergence, the shape of the sinusoid is approximately being reproduced. The noise is large because the input neurons are spiking at reasonably low rates (between 20-180 Hz, see Methods). To confirm that the slow sinusoid is well captured, we use the periodicity of the stimulus to average multiple chunks of two seconds length. When averaged over all periods after convergence, the averaged pattern can be represented as in Figure 6C). Figure 6D shows the convergence of the weights.

Similar to the smoothing performed for the second derivative operator $d_{(2)}$, the control learning rules defined by $\Lambda_{Hebbian+}$ (implementing PCA), $\Lambda_{Hebbian-}$ and $\Lambda_{Classic}$ are adapted to work with spikes. In fact each control learning rule corresponds to a control STDP kernel represented in

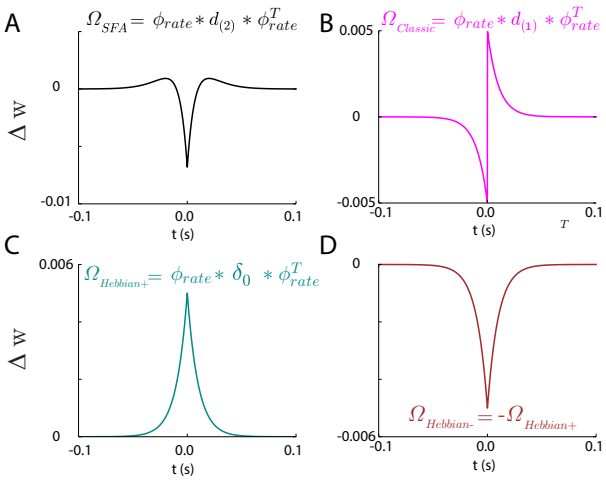

Fig. 7 Representation of the STDP kernels implementing the control learning rules for plasticity (see Methods for literal expressions). **A** SFA kernel as a second derivative. **B** Plasticity kernel based on first order derivative. **C** Hebbian kernel rule implementing PCA [25] when $\tau_{STDP} \to 0$ ms and equivalent to the trace learning [12] in general. **D** Anti-Hebbian kernel.

Figure 7. In particular $\Omega_{Hebbian+}$ is equivalent to trace learning [12] according to our theory and $\Omega_{Classic}$, measured with biological neurons *in vitro* [3] is mostly used in the literature. We show in Figure 8 that only Ω_{SFA} implements SFA for any choice of parameters.

The parameter α controlling the amplitude of the quickly varying components in the toy example is varied for different values of the STDP time constant τ_{STDP}. The influence of α is shown in Figure 8A, while Figure 8B shows the modulation of the plasticity kernel as a function of $\tau_{STDP} = \tau_{rate}$. Figure 8C shows which STDP kernel is capable of recovering the slow sinusoid with an extreme case of a width of $\tau_{STDP} = 0$ ms (left), $\tau_{STDP} = 10$ ms (middle) and $\alpha = 1$, τ_{STDP} being varied (right). Our implementation of SFA recovers the slow solution in each case (see Methods). For large realistic STDP kernel width, the Hebbian kernel only recovers the slow sinusoid for $\alpha < 1000$ and the anti-Hebbian for $\alpha > 1000$. Instead Ω_{SFA} finds the expected solution for any α. When varying τ_{STDP}, there is only one case where Λ_{SFA} does not recover the slow sinusoid: when the plasticity kernel is larger than the fastest oscillations ($\tau_{STDP} = 100$ ms, right panel, last row). But because the assumption of the time scales (3.b, see Methods) is not respected any more, we conclude that the slowest component is robustly extracted by a plasticity kernel shaped by Ω_{SFA} only.

Application to audio recordings

Having shown the validity and the robustness of our implementations on a simple example, we can now use more complex inputs. Because in audio time is crucial and because SFA has mostly been applied to visual stimuli [2, 21, 6], music and speech recordings are good candidates to try our implementations of SFA.

To begin with, we apply SFA on a guitar signal x_0 (15s of a country music), sampled at $f_s = 11$ kHz with our model of a plastic spiking neuron (see Figure 9A for a schematic representation). This one dimensional audio recording is expanded to dimension 64 by delay lines (see Methods), providing multiple delayed versions of x_0, each of them delayed by 0.81 ms compared to the previous one (see Figure 9A). From each channel x_i an inhomogeneous time-varying Poisson spike train θ_i is generated with rates varying between 20 and 180 Hz (see Methods). Using the STDP kernel Ω_{SFA} we see on Figure 9B the convergence of the weights from all those channels when looping over that country guitar recording. Figure 9C displays the final weights assigned to each delay, i.e. the learnt filter given by SFA. We can see that such an optimal filter is a sinusoid at a particular frequency, which corresponds to the dominant frequency of the recording (97Hz being the frequency of G2 the note mostly played by the guitarist).

By using delay lines with larger delays (8.8 ms) between each inputs (see Figure 10A), we can see on Figure 10B the results obtained with the standard SFA algorithm and our implementation. We can see in frequency domain (Figure 10C) that the solutions peak at 3 Hz. Filters, again, are sinusoids (Figure 10D) tuned to about 3 Hz (Figure 10E), i.e. to 180 beats per minute, the tempo of the music piece. Interestingly, the visualisation of the filters in time domain in Figure 10D seems to show that the batch implementation produced a more regular solution than the classic SFA algorithm.

Assuming the length of the delays in the delay lines can impact the filters obtained by SFA algorithm, we ask what is the influence of the parameter τ_{STDP}, the width of the plasticity kernel. In Figure 11, we used the same guitar recording as previously expanded with 512 delays of 1.5 ms, to get a broad range of possible filters. The plasticity kernel has a variable width τ_{STDP} varying between 0.1 ms to about 30 ms. Figure 11A, B, C represents three different learnt filters with different orders of magnitude for τ_{STDP}: 0.1, 1 and 10 ms. Qualitatively the larger the STDP kernel

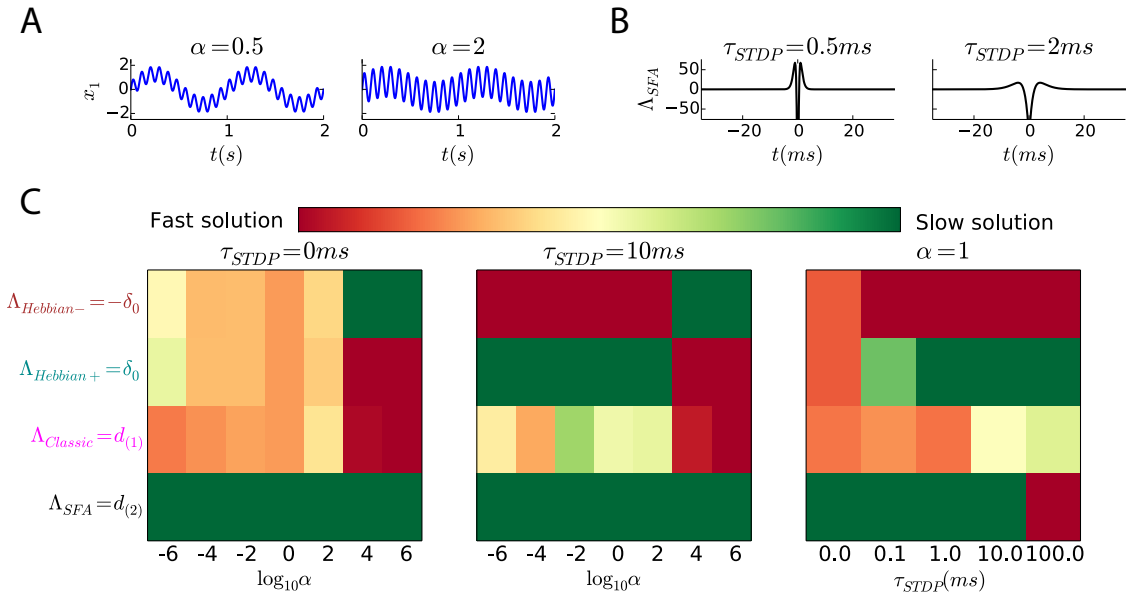

Fig. 8 Robustness to noise and to parameters of the plasticity kernels. **A** Influence of α, balancing the amount of high frequency information in the input (See Methods on the toy example for more details). **B** Influence of τ_{STDP}, the time constant defining the width of the STDP kernel. **C** After fixing one parameter (Left $\tau_{STDP} = 0$ ms, Middle $\tau_{STDP} = 10$ ms, and Right $\alpha = 1$) the other parameter (either τ_{STDP} or α is varying on the horizontal axis). In each case the four STDP kernels introduced in Figure 7 are compared on the vertical axis. For each point of the graphs 20 batch simulations are launched with different random weight initialisation, and slowness is measured as a correlation with the slow sinusoid averaged over trials (see Methods).

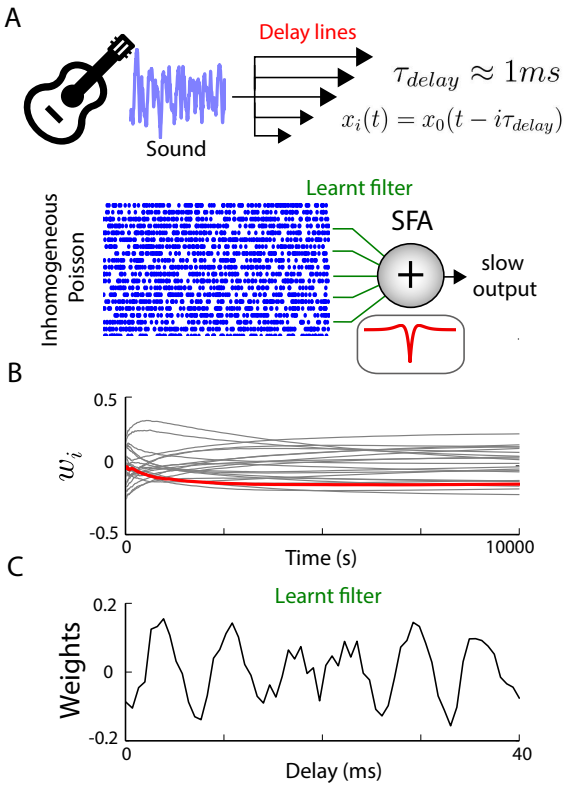

Fig. 9 Illustration of the SFA algorithm and Spike-Timing-Dependent plasticity. **A** Schematic construction of the inputs. A single guitar recording x_0 of country music is delayed $N = 64$ times into channels $x_i(t)$ with a delay being a multiple of $\tau_{delay} = 0.74$ ms (delay line). Every input line is then turned into spikes by inhomogeneous Poisson sources (see Methods) and fed into the SFA algorithm. **B** Evolution of the synaptic weights as function of time. Grey lines show 20 individual trajectories, while the red line shows the sum, proving the convergence. **C** Final weight as a function of the delays. A sinusoidal-like filter has been learnt.

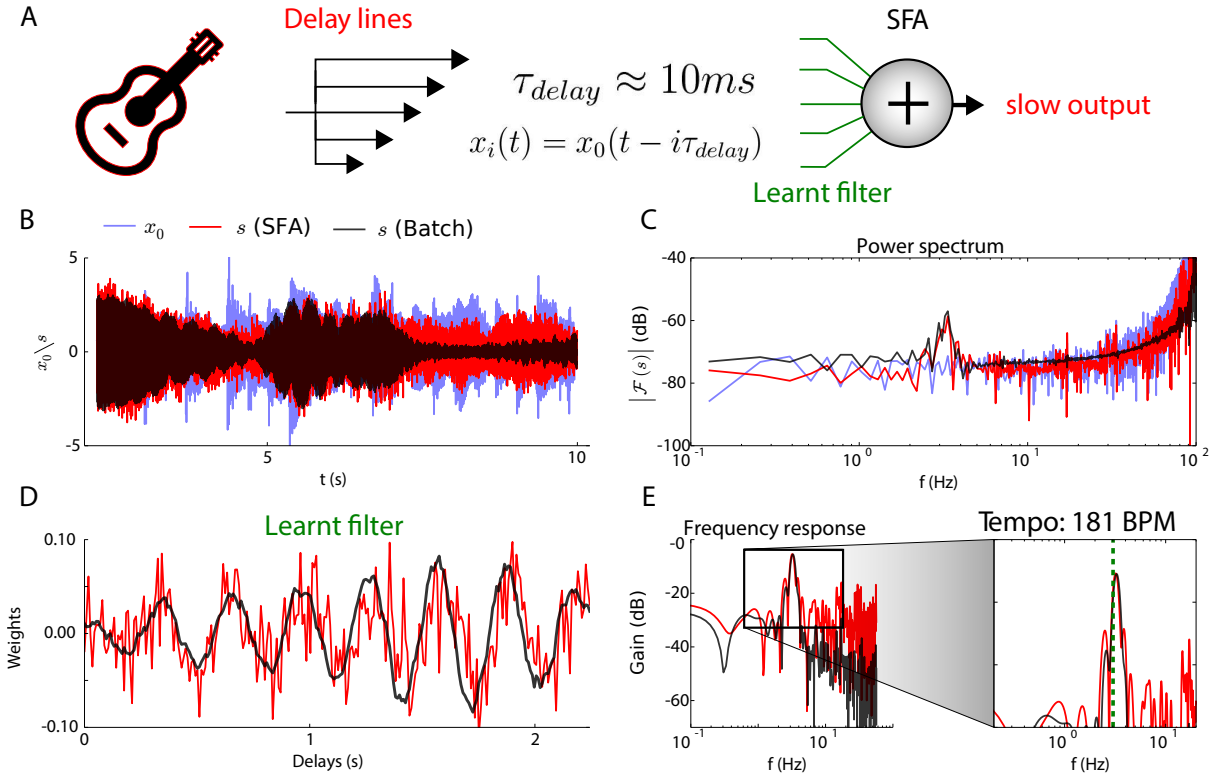

Fig. 10 Extracting the tempo of a guitar signal. **A** Schematic construction of the inputs, as in Figure 9. A single guitar recording x_0 of country music is delayed 256 times into channels $x_i(t)$ with delays of multiple of $\tau_{delay} = 8.8$ ms (delay line), and fed into the SFA algorithm. **B** The audio recording x_0 (blue), compared with the two outputs of SFA (filtered versions of input). In black, the output is obtained with our gradient based implementation of SFA, and in red, this is the original algorithm (ground-truth). **C** Comparison of the power spectrum for the three signals, showing an increase in energy at the tempo frequency (3 Hz). **D** Weights after learning for all input channels with the two implementations of SFA. **E** The power spectrum of the two learnt filters. They are tuned to 3 Hz, the tempo of the recording.

is, the lower the captured frequencies are. Figure 11D represents, in a more exhaustive manner, the frequency responses as a function of τ_{STDP}. Only few frequencies have high energy in the learnt frequency response, and the larger the kernel is, the lower the tuning frequencies of the filter are. For sharp STDP kernel ($\tau_{rate} \approx \delta t$) the pitch G2 is captured. With a slightly larger one the filter is tuned at the pitch E2 (another bass note played by the musician). For wide STDP kernel, the tempo is captured even with small delays between the inputs. In between those note pitches and the tempo, the captured frequencies are more complex to interpret.

Discussion

In this work, we developed a mathematical framework showing how SFA could be implemented by models of neurons. We first designed an online learning rule to extract the slowest components of their inputs, and then adapted it in the context of spiking neurons. It provides a neural implementation of SFA compatible with the experimental observation of Spike-Time-Dependent-Plasticity [3, 16] and rate homoeostasis. By implementing a gradient descent on the cost function of SFA, we found the precise plasticity kernel that fits a particular function. We showed through simulations that the STDP kernel has to be shaped as a reversed Mexican hat (second derivative) to learn the optimally slow features. When applied to auditory signals, simulations showed that our gradient implementation of SFA allows a single neuron to extract the tempo of its inputs.

The compatibility of SFA and STDP was already demonstrated in [29]. With continuous and deterministic inputs they characterized that with any filter Λ having a Fourier transform of the form $-f^2$ (f is the frequency) truncated to be zero at frequencies higher than $f_{\max}$, a gradient descent of the form $\Delta w_k \propto \langle x * \Lambda, s \rangle$ implements SFA when the input information does not contain frequencies higher than $f_{\max}$. Because the second derivative is a convolution by a filter being $-f^2$ in Fourier domain, our analysis is strictly equivalent

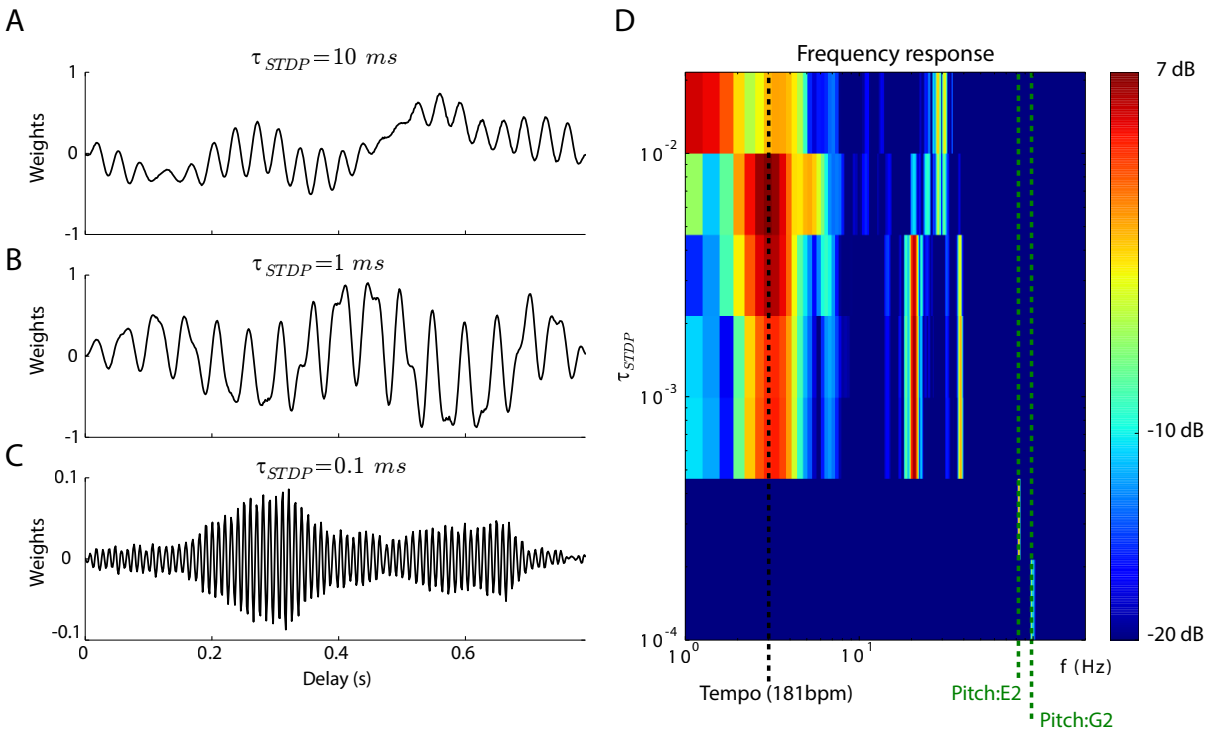

Fig. 11 Influence of the width of the plasticity kernel τ_{STDP}. Simulations are run on the guitar example (see Figure 10) with 512 delays and $\tau_{delay} = 1.5$ ms. Over the trials the dependency to the parameter τ_{STDP} is explored with the batch implementation. **A, B, C** Three filters learnt with τ_{STDP} at different orders of magnitude, from top to bottom $\tau_{STDP} = 10$ ms, $\tau_{STDP} = 1$ ms, $\tau_{STDP} = 0.1$ ms respectively. The larger τ_{STDP} the lower the captured periodicity. **D** Power spectrum of the learnt filters, as a function of the width of the plasticity kernel, τ_{STDP}, varied on the vertical axis. Each row is the frequency response of the learnt filter by SFA, color-coded. Dashed lines annotate the different identified peaks of the frequency response. In green, E2 and G2 are the two bass notes mostly played by the musician in the recording.

but does not require to limit the input frequencies below f_{max}.

In this work, various kernels have been tested as controls, and among those, the classical STDP kernel $\Omega_{Classic}$ and the trace learning equivalent to a plasticity kernel using $\Omega_{Hebbian+}$. In agreement and as suggested in [29], we found that the trace learning rule is a good approximation of SFA (see performances on Figure 8C) for simulations on several toy examples. However, this control kernel does not extract the slow feature any more when a lot of energy is contained in the high frequencies. Control simulations with $\Omega_{classic}$ and the insight of our implementation with a second order derivative allow to raise the problem that the classical asymmetric STDP described in [3] is not suited for SFA. The classical asymmetric STDP kernel was supported in [29], and the divergence comes from the fact that they used PSP time constants much larger than τ_{rate} when we did the opposite. In the limit of large ξ, we also find[3] an asymmetric kernel but it is a smoothed third order derivative which is not as trivial as the classic STDP.

In all the paper, the expression of the SFA problem is simplified assuming inputs have been already pre-processed and decorrelated. Various theories may explain how inputs onto sensory cortices can be decorrelated [26, 7], but a plausible implementation of such a process with spiking neurons is beyond the scope of this manuscript. In addition, a clear restriction here is that in the generic formulation of SFA, a decorrelation process at the output stage is required to provide a rich population of non-redundant SFA neurons. However, we believe that such a decomposition could be mediated with lateral inhibition and competition between several output neurons [11].

[3] As done in [29] we searched for the STDP kernel that implements SFA by verifying $d_{(2)} = \Omega * \xi^T$ with ξ very large. In this limit, to invert the convolution by ξ we can perform a derivation (see [29]) so that the solution for Ω is a third order derivative.

Appendix

Filtered auto-correlation for a Poisson spike train The main difference between the implementation of SFA with a spiking neuron and with a linear-rate-based neuron raises from the fact that the expected value of the auto-correlation of a Poisson spike train is not the auto-correlation of the rates; an additional term appears. This is justified in [22] and used in [29]. Formally, for any Poisson spike train θ of rate ν, time t, given two filters f and g, we have if $\mathbb{E}(x)$ stands for the stochastic average over multiple realizations of a random variable x

$$\mathbb{E}\left[\langle\theta * f, \theta * g\rangle\right] = \langle\nu * f, \nu * g\rangle + \overline{\nu}\,\overline{fg} \tag{12}$$

Equation (12) is obtained because we have $\forall u, v \mathbb{E}\left[\theta(u)\theta(v)\right] = \nu(u)\nu(v) + \delta_u(v)\nu(u)$ because spiking at time u and v are dependent only if u equals v for which the variance of $\theta(u)$ is equal to its mean $\nu(u)$. Switching filtering integration and ensemble integration we have $\mathbb{E}\left[\theta * f(t)\theta * g(t)\right] = \int_u \int_v g(t-u) f(t-v) \mathbb{E}\left[\theta(u)\theta(v)\right] dvdu$. Inserting the previous equality leads to

$$\mathbb{E}\left[\theta * f(t)\theta(t)\right] = \nu * f(t)\nu * g(t) + \nu * (fg)(t)$$

and integration over time using the fact that $\overline{\nu * (fg)} = \overline{\nu}\,\overline{fg}$ gives Equation (12).

Average effect of STDP for spiking neurons From Equation (9) we can write the net effect of STDP

$$\langle\dot{w}\rangle = \epsilon\langle\theta_i * \Omega, \theta_{out}\rangle$$

To compute the stochastic average of STDP over multiple realizations of the spike trains, we use $\mathbb{E}\left[\langle\dot{w}\rangle\right] = \epsilon\mathbb{E}\left[\mathbb{E}\left[\langle\dot{w}\rangle|\theta_i\right]\right]$. In other words, for a fixed set of input spike trains θ_i we first integrate over all the possible output spike trains, we insert the linear dynamics of the neuron in Equation (2) and only terms constant in θ remain

$$\begin{aligned}\mathbb{E}\left[\langle\dot{w_i}\rangle|\theta_i\right] &= \epsilon\mathbb{E}\left[\langle\theta * \Omega, \theta_{out}\rangle|\theta_i\right] \\ &= \epsilon\kappa\sum_j w_j\langle\theta_i * \Omega, \theta_j * \xi\rangle + \epsilon\nu_0\overline{\theta_i * \Omega}\end{aligned}$$

Averaging now over all the possible input spike trains, θ_i and θ_j being independent for $i \neq j$, we can replace θ_i and θ_j by their expected value ν_i and ν_j. For the remaining term $i = j$ we use Equation (12), such that we have

$$\begin{aligned}\mathbb{E}\left[\langle\dot{w_i}\rangle\right] = \epsilon\kappa\sum_{j=1}^{n} w_j\langle\nu_i * \Omega, \nu_j * \xi\rangle \\ +\epsilon\kappa w_i r\overline{\Omega\xi} + \epsilon\nu_0 r\overline{\Omega}\end{aligned} \tag{13}$$

The third term can be neglected as in [29], in our case $\overline{\Omega}$ is null anyway. The second remaining term is proportional to w, and this is equivalent to simply adding in the cost function a term proportional to $\|w\|^2$ which would be made obsolete by the weight normalization. What remains for the effect of STDP on average is therefore proportional to $\sum_j w_j\langle\nu_i * \Omega, \nu_j * \xi\rangle$ which can be rewritten as $\langle\nu_i * \Omega * \xi^T, \sum_j w_j\nu_j\rangle$.

Finally, to rigorously show that this is equivalent to the gradient implementation of SFA with continuous neurons obtained in Equation 8, we replace ν by $r + r_s x$ in Equation 13 and we obtain

$$\begin{aligned}\mathbb{E}\left[\langle\dot{w_i}\rangle\right] = \epsilon\kappa r_s^2\langle x_i * \Omega * \xi^T, \sum_j w_j x_j\rangle \\ +\epsilon\kappa r_s^2\langle x_i * \Omega * \xi^T, r w_{sum}\rangle \\ +\epsilon\kappa r_s^2\langle r\overline{\Omega}\,\overline{\xi^T}, \sum_j w_j\nu_j\rangle\end{aligned}$$

The second term is null because it is equivalent to integrating a filtered version of a derivative of x over time and x is null at infinity, and the third term is null because $\overline{\Omega}$ is null. What remains is therefore only

$$\mathbb{E}\left[\langle\dot{w_i}\rangle\right] = \epsilon\kappa r_s^2\langle x_i * \Omega * \xi^T, s\rangle \tag{14}$$

References

1. S Becker and G E Hinton. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355:161–163, 1992.
2. Pietro Berkes and Laurenz Wiskott. Slow Feature Analysis yields a rich repertoire of complex cell properties. *J Vis*, 5(6):579–602, July 2005.
3. G Q Bi and M M Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, 18:10464–10472, 1998.
4. T V P Bliss and T Lomo. Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the preforant path. *Journal of physiology*, 232:331–356, 1973.
5. Claudia Clopath, Lars Büsing, Eleni Vasilaki, and Wulfram Gerstner. Connectivity reflects coding: a model of voltage-based STDP with homeostasis. *Nature neuroscience*, 13(3):344–52, March 2010.
6. Sven Dähne, Niko Wilbert, and Laurenz Wiskott. Slow Feature Analysis on retinal

waves leads to V1 complex cells. *PLoS computational biology*, 10(5):e1003564, May 2014.

7. Y Dan, J J Atick, and R C Reid. Efficient coding of natural scenes in the lateral geniculate nucleus: experimental test of a computational theory. *The Journal of neuroscience*, 16:3351–62, 1996.
8. Yang Dan and Mu-ming Poo. Spike Timing-Dependent Plasticity of Neural Circuits. *Neuron*, 44(1):23–30, September 2004.
9. James J DiCarlo, Davide Zoccolan, and Nicole C Rust. How Does the Brain Solve Visual Object Recognition? *Neuron*, 73(3):415–434, February 2012.
10. Wolfgang Einhäuser, Jörg Hipp, Julian Eggert, Edgar Körner, and Peter König. Learning viewpoint invariant object representations using a temporal coherence principle. *Biological Cybernetics*, 93:79–90, 2005.
11. Michael S Falconbridge, Robert L Stamps, and David R Badcock. A simple Hebbian/anti-Hebbian network learns the sparse, independent components of natural images. *Neural Computation*, 18(2):415–429, February 2006.
12. P Földiák. Learning invariance from transformation sequences. *Neural Computation*, 200:194–200, 1991.
13. Mathias Franzius, Henning Sprekeler, and Laurenz Wiskott. Slowness and sparseness lead to place, head-direction, and spatial-view cells. *PLoS Computational Biology*, 3(8):e166, 2007.
14. Mathieu Galtier and Gilles Wainrib. Multiscale analysis of slow-fast neuronal learning models with noise. *Journal of Mathematical Neuroscience*, 2:13, November 2012.
15. Mathieu N Galtier and Gilles Wainrib. A biological gradient descent for prediction through a combination of stdp and homeostatic plasticity. *Neural computation*, 25(11):2815–2832, 2013.
16. Wulfram Gerstner, Richard Kempter, J Leo van Hemmen, Hermann Wagner, and JL Van Hemmen. A neuronal learning rule for sub-millisecond temporal coding. *Nature*, 383(2):76–78, 1996.
17. J J Gibson. The Ecological Approach to Visual Perception., 1986.
18. Dan Goodman and Romain Brette. Brian: A Simulator for Spiking Neural Networks in Python. *Frontiers in Neuroinformatics*, 2, November 2008.
19. DO O Hebb. The organization of behavior: a neuropsychological theory. *Science Education*, 44:335, 1949.
20. Eugene M Izhikevich and Niraj S Desai. Relating STDP to BCM. *Neural Comput*, 15(7):1511–1523, July 2003.
21. Christoph Kayser, Wolfgang Einhäuser, Olaf Dümmer, Peter König, and Konrad P Körding. Extracting Slow Subspaces from Natural Videos Leads to Complex Cells. In *Artificial Neural Networks - ICANN 2001*, volume 2130, pages 1075–1080, 2001.
22. Richard Kempter, Wulfram Gerstner, and J Leo Van Hemmen. Hebbian learning and spiking neurons. *Physical Review E*, 59(4):4498, 1999.
23. H. Markram, J Lübke, M Frotscher, and B Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, 275(5297):213–5, January 1997.
24. D Marr. A theory for cerebral neocortex. *Proceedings of the Royal Society of London. Series B*, 176:161–234, 1970.
25. Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.
26. Christian Pozzorini, Richard Naud, Skander Mensi, and Wulfram Gerstner. Temporal whitening by power-law adaptation in neocortical neurons. *Nature neuroscience*, 16(7):942–8, July 2013.
27. R Quian Quiroga, L Reddy, G Kreiman, C Koch, and I Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435(7045):1102–7, June 2005.
28. R P Rao and T J Sejnowski. Spike-timing-dependent Hebbian plasticity as temporal difference learning. *Neural Computation*, 13(10):2221–2237, October 2001.
29. Henning Sprekeler, Christian Michaelis, and Laurenz Wiskott. Slowness: An objective for Spike-Timing-Dependent Plasticity? *PLoS Computational Biology*, 3(6):e112, 2007.
30. Ian H. Stevenson, Beau Cronin, Mriganka Sur, and Konrad P. Kording. Sensory adaptation and short term plasticity as bayesian correction for a changing brain. *PLoS ONE*, 5(8), 2010.
31. Taro Toyoizumi, Jean-Pascal Pfister, Kazuyuki Aihara, and Wulfram Gerstner. Optimality Model of Unsupervised Spike-Timing-Dependent Plasticity: Synaptic Memory and Weight Distribution. *Neural Computation*, 19(3):639–671, March 2007.
32. Richard Turner and Maneesh Sahani. A maximum-likelihood interpretation for Slow Feature Analysis. *Neural computation*, 19(4):1022–38, April 2007.

33. G Wallis and E T Rolls. Invariant face and object recognition in the visual system. *Progress in neurobiology*, 51:167–194, 1997.
34. Laurenz Wiskott and Pietro Berkes. Is slowness a learning principle of the visual cortex? *Zoology (Jena, Germany)*, 106(4):373–82, jan 2003.
35. Laurenz Wiskott and Terrence J Sejnowski. Slow Feature Analysis: unsupervised learning of invariances. *Neural Comput*, 14(4):715–770, April 2002.
36. Pierre Yger, Sami El Boustani, Yves Frégnac, Alain Destexhe, and S El Boustani. Stable learning in stochastic network states. *Journal of neuroscience*, 32(1):194–214, January 2012.
37. Pierre Yger and Kenneth D. Harris. The Convallis rule for unsupervised learning in cortical networks. *PLoS Computational Biology*, 9(10):1–32, October 2013.