frontiers in
COMPUTATIONAL NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 19 November 2012
doi: 10.3389/fncom.2012.00078

Support vector machines for spike pattern classification with a leaky integrate-and-fire neuron

Maxime Ambard [1,2] **** and Stefan Rotter*** [1,2]

[1] *Bernstein Center Freiburg, University of Freiburg, Freiburg, Germany*
[2] *Faculty of Biology, University of Freiburg, Freiburg, Germany*

Edited by:
David Hansel, University of Paris, France

Reviewed by:
Germán Mato, Centro Atomico Bariloche, Argentina
Carl Van Vreeswijk, CNRS, France

****Correspondence:***
Maxime Ambard, Bernstein Center Freiburg, University of Freiburg, Hansastr. 9a, 79104 Freiburg, Germany.
e-mail: maxime.ambard@bcf.uni-freiburg.de

Spike pattern classification is a key topic in machine learning, computational neuroscience, and electronic device design. Here, we offer a new supervised learning rule based on Support Vector Machines (SVM) to determine the synaptic weights of a leaky integrate-and-fire (LIF) neuron model for spike pattern classification. We compare classification performance between this algorithm and other methods sharing the same conceptual framework. We consider the effect of postsynaptic potential (PSP) kernel dynamics on patterns separability, and we propose an extension of the method to decrease computational load. The algorithm performs well in generalization tasks. We show that the peak value of spike patterns separability depends on a relation between PSP dynamics and spike pattern duration, and we propose a particular kernel that is well-suited for fast computations and electronic implementations.

Keywords: machine learning, synaptic kernel, supervised learning rule, Tempotron, linear separation

1. INTRODUCTION

Spike pattern classification has become an important topic in several fields of research. For example, electrophysiological recordings from neural networks on Multi-Electrode Arrays (MEA) provide spatio-temporal spike patterns that remain difficult to understand. In those setups, spike pattern classification algorithms are useful to understand how spontaneous network bursting activities that propagate through the neural network are triggered in particular places (Kermany et al., 2010). Experimental evidence tends to show that precise spike timing plays a significant role in the encoding of sensory stimuli (Bohte, 2004; VanRullen et al., 2005), and theoretical considerations have shown that spiking neuron models can have better computational capabilities for fast information processing than firing rate based neuron models (Maas, 1997). Understanding how synaptic weights determine the neuronal input–output function is one of the most important challenges of computational neuroscience. Electronic devices based on temporal pulse coding have interesting properties for sensors and neuroprosthesis development (Boahen, 2002; Ambard et al., 2008; Chen et al., 2011), and recent research on novel computer architectures focuses on low-power spike based computing chips (Modha, 2011).

Previous methods for spike pattern classification can be separated into two families. The first family uses spike train metrics (for an overview, see Brown et al., 2004; Grün and Rotter, 2010). Some of these methods are based on arbitrary features extracted from the observed spike patterns such as rank order (Pan et al., 2009), firing rate (Kermany et al., 2010), or inter-spike intervals (ISI) (Abeles and Gerstein, 1988; Christen et al., 2004). These methods run the risk of preselecting features that are not relevant for biology. Others use metrics based on vector-space embeddings (van Rossum, 2001; Schrauwen and Campenhout, 2007; Houghton and Sen, 2008). While those methods are well-known and show good performance in classification tasks, their results are difficult to understand in terms of neural mechanisms. The second family is bio-inspired. It uses neuron models and sets appropriate synaptic weights. Classification is done with a supervised learning rule to control the neural response for a given set of spike patterns (Bohte et al., 2000; Gütig and Sompolinsky, 2006; Kasiński and Ponulak, 2006; Voegtlin, 2007; Mc Kennoch et al., 2009; Urbanczik and Senn, 2009). Those methods are particularly interesting since their results can be easily linked to biological processes.

In this paper, we present a method related to both families of methods. It is based on spike train kernel convolution in the time domain, in conjunction with Support Vector Machines (SVM) to compute an optimal linear decision policy. Coefficients of the corresponding hyperplane can be interpreted as synaptic weights on the dendrite of a leaky integrate-and-fire (LIF) neuron model. In section 2 we describe our new method. We compare its generalization performance with other learning rules sharing the same conceptual framework. We show a relation between spike pattern separability and postsynaptic potentials (PSP) kernel dynamics, and we introduce kernels optimized for fast computation. Section 3 describes in detail the methods used in the simulations. In section 4 we discuss limitations and possible improvements of our approach.

2. RESULTS

2.1. SUPPORT VECTOR MACHINE USING POSTSYNAPTIC POTENTIAL KERNELS

Trying to understand how spike patterns are discriminated in biological neural networks by selecting features or by mapping spike trains in *ad-hoc* vector spaces requires assumptions on relevant features used by neural information processing. Another method takes a neuron's point of view by modeling neuronal information

processes and letting the neuron figures out which information of spike patterns is relevant. One of the major aspects of the neural processing is the temporal integration of postsynaptic excitatory and inhibitory events that results in a temporal neural response expressed by action potentials. In this scenario, the neural input–output function is determined by the modulation of synaptic event amplitudes, called synaptic weights in numerical models.

The present section describes a new supervised learning rule that optimizes synaptic weights for robust spike patterns classification. Although we would not characterize this method as biomimetic, it is related to a common LIF neuron model. The results, that can be considered as optimal synaptic weights, provide new insight on what should be the outcome of biological synaptic plasticity processes for robust spike patterns classification. In the present paper, we refer to this method using the acronym SVM-PSP for Support Vector Machine using Post-Synaptic Potential kernels.

We consider two classes of spike patterns $P^+ = \{p^+\}$ (target patterns) and $P^- = \{p^-\}$ (background patterns) that have to be separated from each other. Each spike pattern is composed of incoming spikes from N neurons. We define t_{ij} as the time of the jth spike of the ith neuron. Let $k(t)$ be a fixed kernel. For each time t we consider the vector $f(t) = (f_1(t), f_2(t), \ldots, f_N(t))$ where $f_i(t) = \sum_j k(t - t_{ij})$. Evaluating this function at discrete points in time $t_l = \Delta\, l$, a pattern p is transformed into a set of points $f_p = \{f(t_1)\}$ residing in an N-dimensional feature space $S = \mathbb{R}^N$. Let us denote $F^+ = \{f^+\}$ (resp. $F^- = \{f^-\}$) the set of points associated to all p^+ (resp. p^-) patterns (see **Figures 1A–D** for illustration).

The aim of our classification is to find a hyperplane $H(W, b)$ given by the equation $W_1 f_1 + W_2 f_2 + \ldots + W_N f_N - b = 0$ in S that separates at least one point $f^+(t_l)$ of each f^+ from all the points $f(t_l)$ of all f^- as described by the following equations

$$\forall p \in P^+, \exists t_l : \sum_i W_i \sum_i k(t_l - t_{ij}) - b \geq 0 \qquad (1)$$

$$\forall p \in P^-, \forall t_l : \sum_i W_i \sum_j k(t_l - t_{ij}) - b < 0 \qquad (2)$$

If we consider that only one pattern has to be detected, the classification task is to find at least one point $f(t_l)$ of f^+ that can be linearly separated from all the points $f^-(t_l)$ in F^-.

Linear SVM (Vapnik, 1995) can be used to find a hyperplane that optimally separates two classes of points. We used the solver L2-regularized L1-loss support vector classification (dual) with a cost parameter of 10, epsilon of 10^{-2} and an enabled bias, provided by the open-source liblinear python/C++ library (Fan et al., 2008).

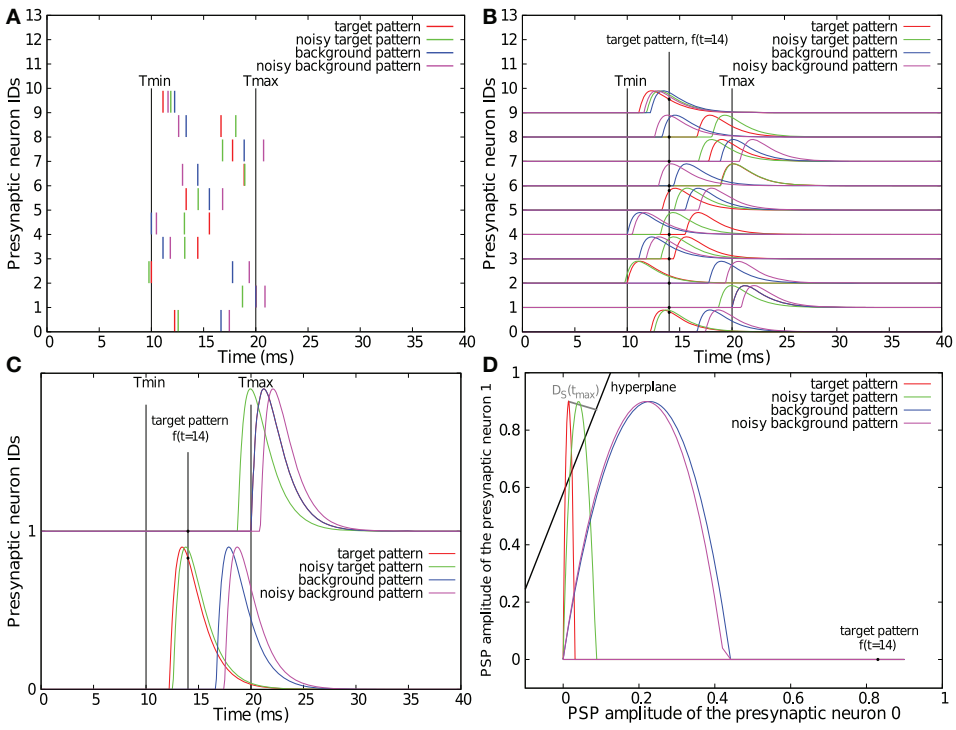

FIGURE 1 | Illustrations of the SVM-PSP method. One target and one background spike pattern, superimposed with their noisy version (standard deviation of the Gaussian noise = 1 ms) as generated for section 2.2 **(A)**. Convolution of the spike patterns with a double exponential kernel with $\tau_r = 1$ ms and $\tau_d = 1.5$ ms. Note that the maximal PSP amplitude has been fixed to 0.9 for illustration purposes **(B)**. Convolution of spike patterns with two input neurons **(C)** and representation of their trajectories $f(t)$ in the 2-dimensional feature space S superimposed with the computed separating hyperplane **(D)**. Note that orthogonal projection $D_S(t_{\max})$, representing the maximal D_S^+ distance onto the hyperplane, does not appear orthogonal due to different horizontal and vertical axis scalings.

Before computing the separating hyperplane, all the points are re-scaled such that the minimal value on each dimension is 0 and the maximal value is 1. To get the separation hyperplane, when only 1 p^+ has to be separated, all the $f^-(t_l)$, $p \in P^-$ are considered as points that should remain below a hyperplane, and all the points $f^+(t_l)$ are tried successively as point that should remain above this hyperplane.

Thus, a new separation hyperplane H_l is computed by the SVM for each tested $f^+(t_l)$ and the aim is to find H_l that can separate one $f^+(t_l)$ from all the $f^+(t_l)$ points with the largest margin. To find this separation hyperplane, let us first consider the signed orthogonal distance between a point $f(t_l)$ and a hyperplane H given by

$$D_S(H, f(t_l)) = \frac{\langle W, f(t_l)\rangle - b}{\|W\|} \tag{3}$$

For each tested $f^+(t_l)$ point, we compute $D_S^+ = D_S(H_l, f^+(t_l))$, its distance from the corresponding hyperplane H_l. The minimal distance between all the $f^-(t_l)$ points and the hyperplane is also computed and is given by

$$D_S^- = -\max(D_S(H_l, f^-(t_l))) \tag{4}$$

The ability of the hyperplane H_l to separate the $f^+(t_l)$ point from all the $f^-(t_l)$ is given by

$$D_S(H_l, f^+(t_l), P^-) = \min(D_S^+, D_S^-) \tag{5}$$

$D_S(H_l, f^+(t_l), P^-)$ is successively computed for all $f^+(t_l) \in p^+$. The hyperplane H_l leading to the maximal margin $D_S(p^+, P^-)$ is considered as the result of our method.

$$D_S(p^+, P^-) = \max(D_S(H_l, f^+(t_l), P^-)) \tag{6}$$

To compute a normalized separability measure that is independent of the number of dimensions of the space S (the number of input neurons), we compute $D_N(p^+, P^-) = 2\,D_S(p^+, P^-)/\sqrt{N}$ where $\sqrt{N}/2$ is the maximal separation margin between two points in normalized feature space $[0, 1]^N$.

Although this learning rule is not bio-mimetic and takes place in a high-dimensional feature space, its result is directly related to LIF neuron parameters. Hyperplane coefficients W can be transformed into synaptic weights w by multiplying them by θ/b where θ is the LIF spike threshold potential. After the best separation hyperplane has been found, the dot product between the vector $\theta/b\,W$ and the vector $f(t_l)$ is equivalent to the LIF membrane voltage at time t_l. The result of this method keeps all the points $f^-(t_l)$ of all background patterns below the hyperplane and at least one point $f^+(t_l)$ of the target pattern above this hyperplane. Therefore, it is equivalent to fixing synaptic weights that constraint the LIF membrane voltage below the threshold potential for background patterns, ensuring that membrane voltage exceeds the threshold at least once for the target pattern. Thus, the neuron fires only when the learned target pattern is received. However, using SVM to maximize D_S not only fixes a separation hyperplane, but it also improves the classification reliability against noisy spike patterns. This property is shown in the next section.

2.2. USING SVM IMPROVES GENERALIZATION PERFORMANCE

To quantify the robustness of our method (see section 2.1) against noisy spike patterns, we compared its generalization performance against the original Tempotron algorithm (Gütig and Sompolinsky, 2006 see section 3.2), and a slightly modified Tempotron learning rule that we call the voltage-margin Tempotron (explained in section 3.3).

We used the double exponential $k(t) = e^{-t/1.5} - e^{-t}$ as kernel function. Five negative patterns and one positive pattern were generated (see section 3.1). Time between 0 and 40 ms was discretized with a period of 0.1 ms. For each noise level, the simulation was repeated over 100 trials. In each trial, 100 noisy patterns were created from each learned pattern (i.e., 100 noisy target patterns, 500 noisy background patterns) and used for testing. The three hyperplanes (H_{SVM}, H_{Temp}, and $H_{\mathrm{Temp_M}}$, respectively) found by applying the SVM-PSP, the Tempotron and the voltage-margin Tempotron algorithms were tested on a generalization task on same noisy spike patterns (see section 3.1).

Figures 2A,**B** show that the original Tempotron learning rule leads to a false negative (FN) error rate of about 30% larger than the voltage-margin Tempotron and the SVM-PSP, while it has a low false positive (FP) error rate. As explained in section 3.2, the original Tempotron learning rule stops when a separation has been found without trying to optimize it. This causes generalization performance limitations, especially with regard to FN error when the synaptic weights are initialized to 0, leading to minimal synaptic weights. This explains why, even for small noise, FN error is large and why FP error is small.

Voltage-margin Tempotron and SVM-PSP learning rules have a similar FN error-rate. The SVM-PSP algorithm provides better FP error rate than the other two algorithms, except for the case of strong noise, where the original tempotron and the SVM-PSP perform similarly. The voltage-margin Tempotron overcomes the synaptic weights initialization problem by increasing the voltage margin between points relatively to the voltage threshold, but results are still lower than SVM-PSP. This shows that maximizing D_S is better than D_V (see sections 2.1 and 3.3) in improving generalization results.

However, while the Tempotron learning rule does not require many changes when the task is to separate several p^+ patterns instead of one, this is more problematic for the SVM-PSP method. Separating n p^+ patterns with this method would require testing all possible n-tuples $[f_1^+(t_{l_1}), \ldots, f_n^+(t_{l_n})]$. Due to combinatorial explosion, it is practically impossible to compute all hyperplanes to find the one leading to the highest separation. Genetic algorithms offer a good solution to this problem when dealing with optimization of non-linear functions in which parameters are subject to combinatorial explosion.

Generalization performance was compared for the separation hyperplanes found with a genetic algorithm combined with the SVM-PSP method (see section 3.4), an original Tempotron (see section 3.2) and a voltage-margin Tempotron (see section 3.3). This was performed for a task consisting in separating 2 p^+ from 4 p^- patterns. To prevent promoting one algorithm against

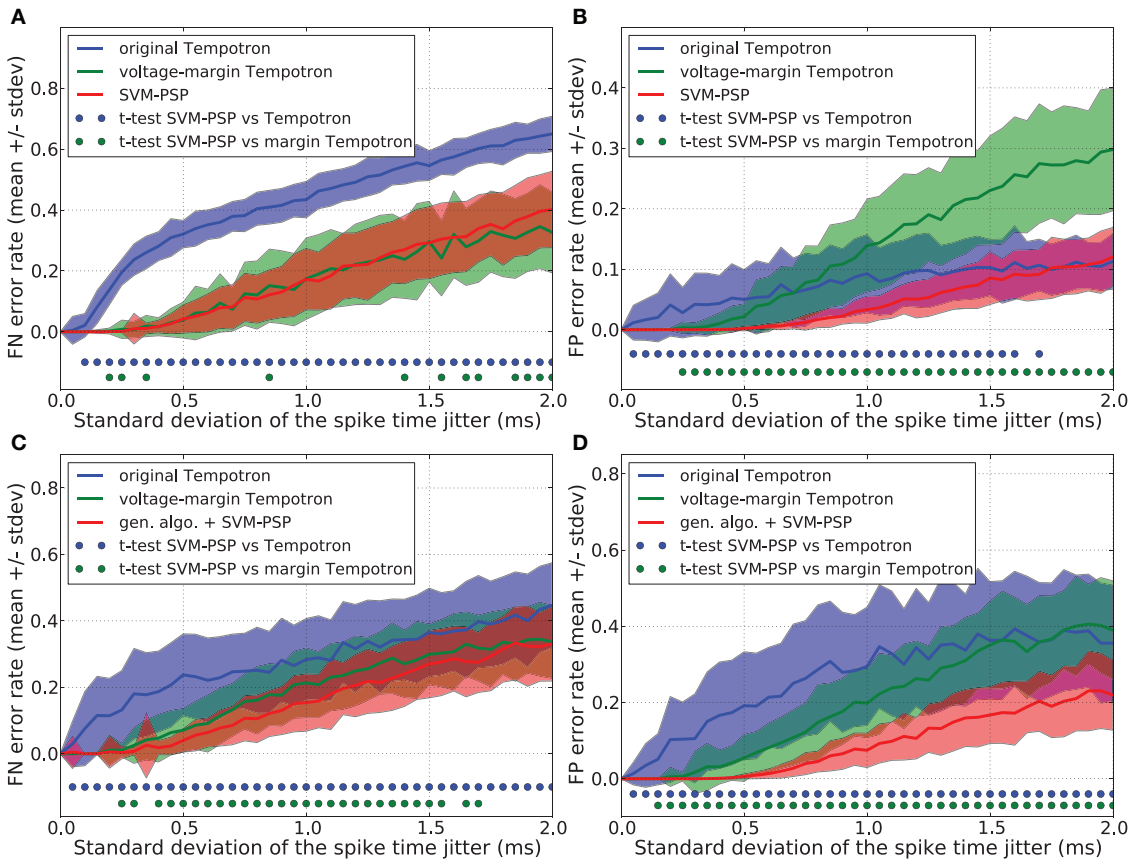

FIGURE 2 | Generalization performance measured as the number of misclassified noisy patterns over the total number of tested noisy patterns. Comparison between the original Tempotron, a voltage-margin Tempotron, and SVM-PSP algorithm with 10 presynaptic neurons. Mean value and standard deviation of false negative (FN) error **(A)** and false positive (FP) error **(B)**. The learning task is to separate one target pattern from 5 background patterns. Hundred jittered variants were generated from each learned pattern and used for testing the generalization abilities of our method. For each noise level, 100 trials were then performed. Mean value and standard deviation of false negative (FN) error **(C)** and false positive (FP) error **(D)**. The learning task is to separate 2 target patterns from 4 background patterns. Blue (resp. green) dots denote a significant difference of the generalization results (paired t-test, $p < 0.05$) between our method and the original Tempotron (resp. voltage-margin Tempotron).

another in terms of learning cycles, we took the number of hyperplanes calculated by the voltage-margin Tempotron learning rule as the maximum number of genotypes tested by the genetic algorithm.

Figures 2C,**D** show the FN and FP error rates. Generalization performance is lower both for the original Tempotron and the voltage-margin Tempotron learning rule as compared to the SVM-PSP method associated with a genetic algorithm. This is especially relevant for FP errors. The original Tempotron has a more symmetric error profile compared to **Figure 2A**. Adding a second target pattern eliminates the influence of null synaptic weights initialization (see section 3.2). The error profile of the SVM-tempotron method is also more symmetric, but FN error rate remains larger than FP error rate. Since, compared to p^- patterns, the separation hyperplane is built on few p^+ points that are most probably situated on an extremity of trajectories, they could be more subjected to variation when jitter is added to the spike pattern. The small error rate for the SVM-PSP method shows that maximizing the separation margin in the feature space provides good generalization performance although not all possible hyperplanes have been tested.

We compared the generalization performance between our new method and the others, based on spike patterns with variable ISIs. With this new pattern generation procedure, each neuron fires once at a time that is uniformly distributed in the interval [10 ms, 20 ms]. The same analysis as above was performed and results are shown in **Figure 3**. Whereas the SVM-PSP method has higher FN error rate than the voltage-margin Tempotron for strong noise, our method presents the best global generalization performance (FN + FP) both in separating 1 p^+ from 5 p^-, and 2 p^+ from 4 p^- patterns.

Employing additional simulations, we checked the benefit of using a genetic algorithm compared to a pure stochastic search where both reproduction and mutation are disabled. In this stochastic search, each new generation is composed of random specimen. We measured the best fitness obtained after each generation for both methods, using a task consisting in separating

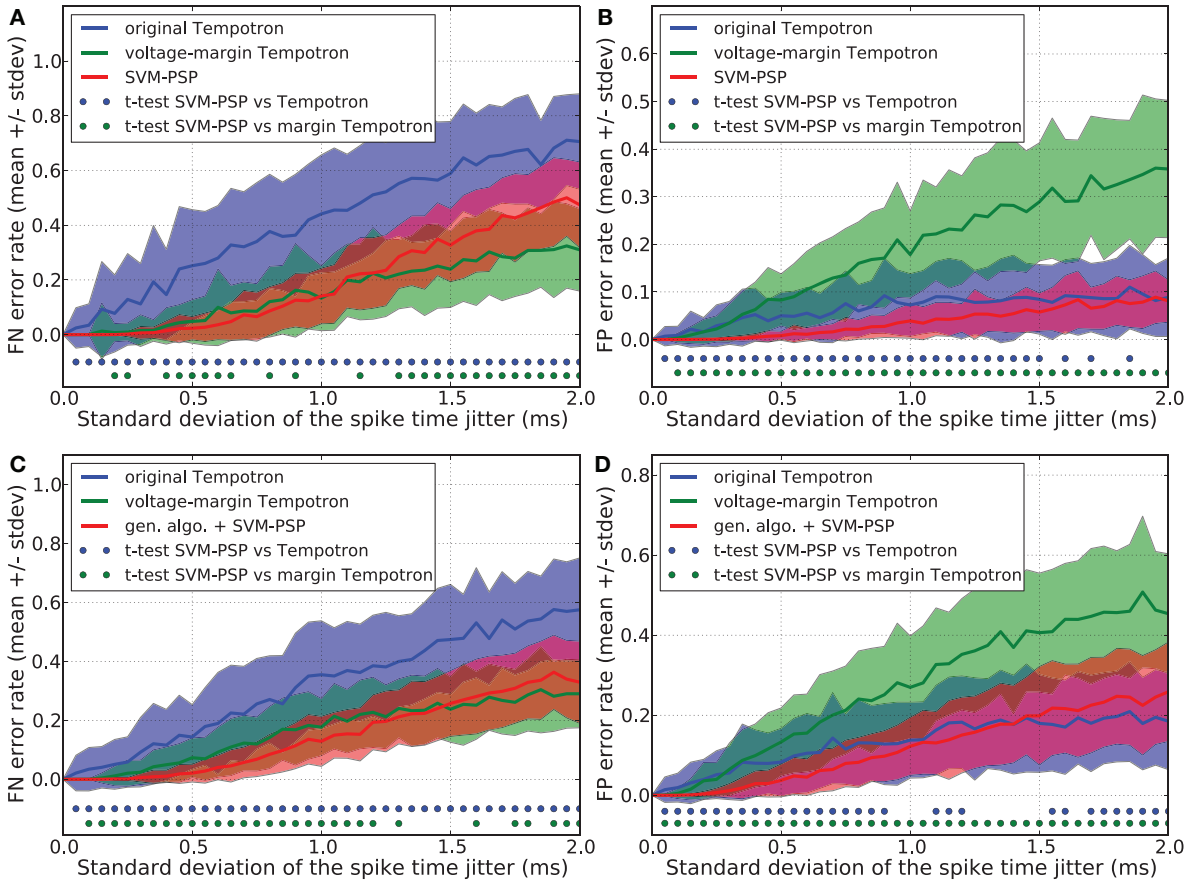

FIGURE 3 | Generalization performance measured as the number of misclassified noisy patterns over the total number of tested noisy patterns. In learned patterns, each neuron fires once at a time that is uniformly distributed in the interval [10 ms, 20 ms]. Comparison between the original Tempotron, a voltage-margin Tempotron, and SVM-PSP algorithm with 10 presynaptic neurons. Mean value and standard deviation of false negative (FN) error **(A)** and false positive (FP) error **(B)**. The learning task is to separate one target pattern from 5 background patterns. Hundred jittered variants were generated from each learned pattern and used for testing the generalization abilities of our method. For each noise level, 100 trials were then performed. Mean value and standard deviation of false negative (FN) error **(C)** and false positive (FP) error **(D)**. The learning task is to separate 2 target patterns from 4 background patterns. Blue (resp. green) dots denote a significant difference of the generalization results (paired t-test, $p < 0.05$) between our method and the original Tempotron (resp. voltage-margin Tempotron).

2 p^+ patterns from 5 p^- patterns, and for another task consisting in separating 3 p^+ patterns from 6 p^- patterns. The simulation was repeated for 1000 trials. Mean and standard deviation of the best fitness are shown in **Figure 4**. Statistical testing was done using a paired t-test for each generation. Compared to a pure stochastic search, the use of a genetic algorithm guarantees a significant improvement which increases with task difficulty.

These results show that using SVM to find a separation hyperplane provides better generalization results than using the Tempotron learning rule, while keeping the same bio-inspired model (i.e., synaptic weights for a simple integrate-and-fire spiking neuron model). Although the use of SVM ensures an optimized linear separation in the feature space S, the shape of the kernel, in particular its time constant, is important in the pre-mapping of spike patterns in the feature space. The next section study this relation.

2.3. SPIKE PATTERN SEPARABILITY DEPENDS ON SHAPE OF POSTSYNAPTIC POTENTIALS

To know how dynamics of the kernel affect the performance of the method, we tested spike pattern separability (see section 2.1) with different kernel functions and various time constants.

Three kernel functions were compared: a single exponential of equation $e^{-t/\tau}$, an α-function $t/\tau\, e^{-t/\tau}$ and a double exponential kernel $e^{-t/\tau} - e^{-t/\tau_r}$ fitted from Povysheva et al. (2006) where the rising time constant τ_r is set to $0.09\,\tau$ (see **Figure 5**).

Spike patterns were generated by the procedure explained in section 3.1. The time constant τ was varied between 0.25 and 64 ms. Time between 0 ms and $\tau + 10$ ms was discretely sampled with resolution of 0.1 ms. The best hyperplane was chosen according to the method described in section 2.1. We used the normalized distance D_N (see section 2.1) to obtain a measure for patterns separability. We write $\nu = \tau/T$ for the time constant normalized for pattern duration, T being the period of learned spike

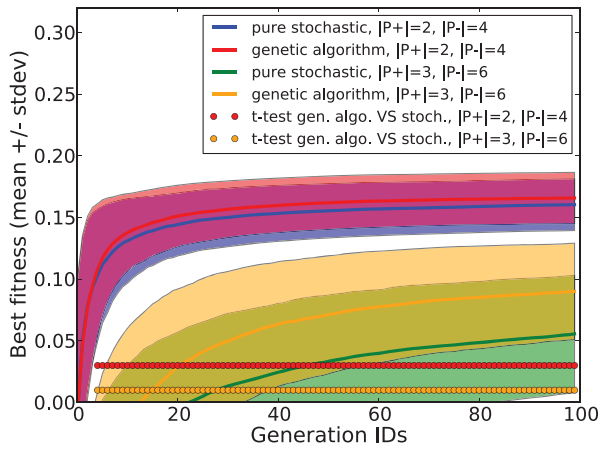

FIGURE 4 | Fitness performances obtained with a genetic algorithm and with a pure stochastic search in tasks consisting in separating 2 p^+ patterns from 4 p^- patterns and separating 3 p^+ from 6 p^- patterns.

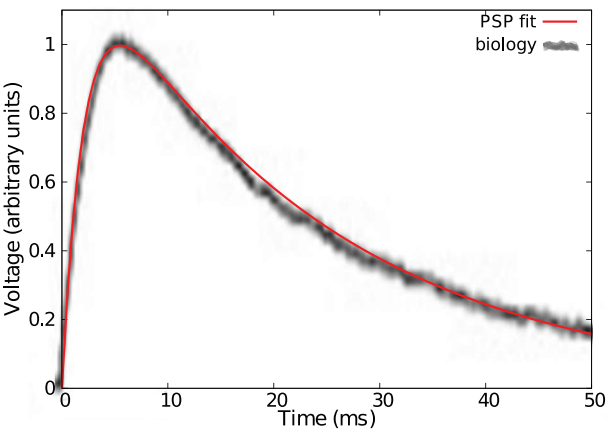

FIGURE 5 | Fit of a typical cortical EPSP with a double exponential bio-mimetic kernel ($\tau = \tau_d = 23$ ms, $\tau_r = 0.09\tau$).

patterns. Maximal distance in the feature space between the trajectory and the synchrony vector $f_s = (1, 1, \ldots, 1)$ was quantified by the measure L_s (see **Appendix A.1**).

Figures 6A–C show the separability for the three kernels, each with spike patterns coming from either 32 or 512 afferent neurons. The task was to separate 1 p^+ pattern from 1 p^- pattern.

For the three tested kernels, separability is low for small ν. Due to the small time constant τ compared to the mean ISI of the spike pattern defined by T/N, all trajectories $f(t)$ fluctuate in the regime of asynchronous inputs, spanned by asynchrony base $B_A = [(0, 0, \ldots, 0), (1, 0, \ldots, 0), \ldots, (0, 0, \ldots, 1)]$. Therefore, f^+ and f^- trajectories remain close to each other, leading to a low separability. Increasing the number of presynaptic neurons from 32 to 512 improves the spike pattern separability, since the mean ISI decreases with the number of afferent neurons. As a result, for a given PSP time constant, the neuron model is better at separating long lasting spike patterns when the number of afferent neurons is large.

Separability for both the α and the double exponential kernels decreases after a peak for intermediate ν values. For large ν values, all $f(t)$ trajectories converge toward the synchrony vector f_s, due to the slow kernel dynamics compared to the ISI of the pattern. At this range of ν values, increasing the number of presynaptic neurons does not affect separability. As a consequence, for all ν values, separability between two spike patterns is better with a large number of afferent neurons.

However, separability with the single exponential kernel increases monotonically with ν. The single-exponential kernel has a discontinuity bringing it instantly from 0 to its maximal value when a spike is received. For large time constants, this kernel becomes similar to a step function. For this reason, the trajectory does not lie on the synchrony vector, but goes from one corner of feature space to another. It reaches the synchrony point f_s when all spikes have been received.

For a simple task such as separating two patterns from each other, separability is closely related to the distance L_s between trajectories and the synchrony vector, especially for large number of presynaptic neurons and large time constants. However, as shown in **Figure 6D**, increasing the number of background patterns to 20 decreases separability, especially when the number of presynaptic neurons is small. For large enough number of afferent neuron compared to the number of background patterns, the volume of the feature space $[0, 1]^N$ is wide enough to provide separability close to L_s. When the number of background patterns increases, the space becomes more "populated" and separability decreases.

Whatever the number of afferent neurons and the number of spike patterns are, the value of ν leading to the separability peak remains the same and can directly be fixed by searching for kernel dynamics that maximize L_s. Therefore, using the measure L_s can give a fast and simple method to properly fix a kernel shape that maximizes spike patterns separability.

Note in **Figure 6C** that a bio-mimetic kernel has its separability peak value for $\nu_{max} = 1.3$. Therefore, maximal pattern separability with a time constant of $\tau = 23$ ms (see **Figure 5**) occurs with a spiking period of $T = \tau\nu_{max} = 17.7$ ms, that corresponds to the low range ($\approx$0.50 Hz) of the so-called (low) γ frequency band (Fries, 2009).

The choice of the PSP kernel function is a key step in designing the algorithm for spike pattern classification. Both too narrow and too wide kernels lead to sub-optimal separability. For one and the same kernel, a neuron receiving input from a larger number of presynaptic neurons is better in separating long-lasting spike patterns. The relation between kernel dynamics and pattern duration for spike pattern classification has already been emphasized using different methods in Rubin et al. (2010).

2.4. PSP KERNELS FOR FAST COMPUTATION WITH THE SVM-PSP METHOD

Convolving incoming spikes with an α-function or a double exponential kernel, such those as presented in sections 2.2 and 2.3,

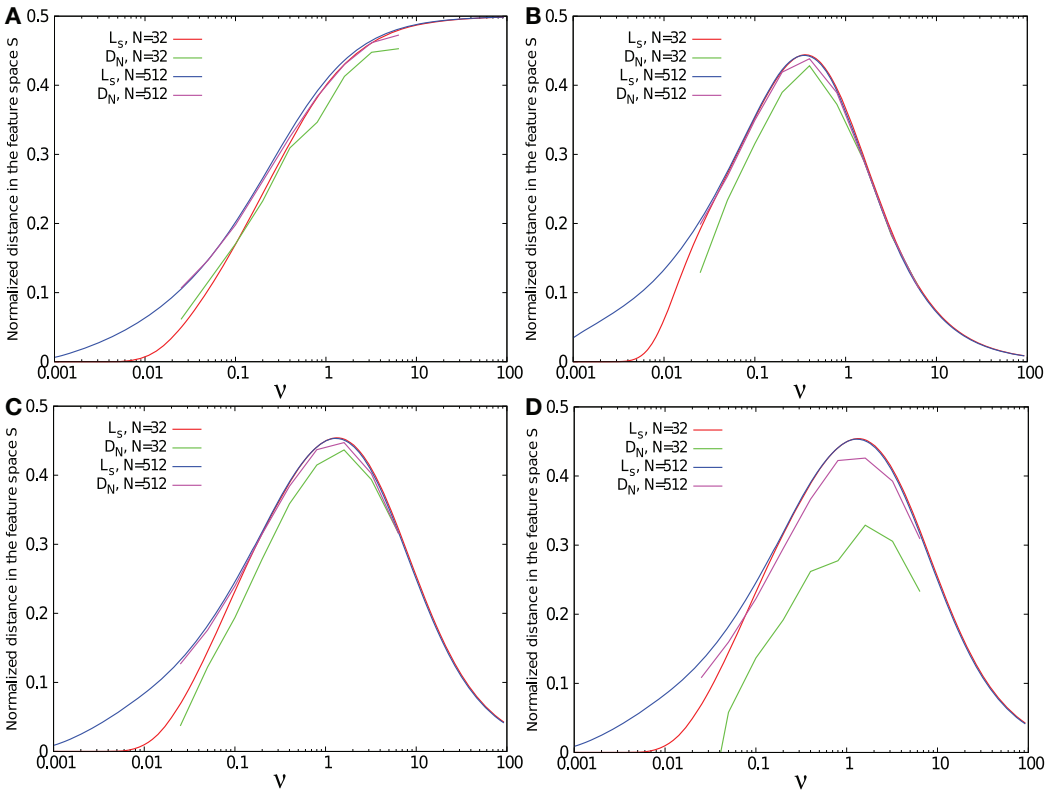

FIGURE 6 | Separability measure $D_N(t_{max})$ and maximal standard deviation L_s as function of the relative time constant of the kernel $\nu = \tau/T$. Results were obtained for a task consisting in separating 1 target pattern from 1 background pattern except in **(D)** where the task was to separate 1 target pattern from 20 background patterns. Simulations were performed using 32 and 512 input neurons, with a single exponential kernel **(A)**, with a α kernel **(B)**, and with a bio-mimetic kernel **(C)** and **(D)**.

leads to curved trajectories in the feature space S. Thus, with our method, every point in the trajectory makes a relevant contribution to find the best separation hyperplane, so a fine temporal discretization of the trajectories $f(t)$ is helpful. However, some PSP kernels have an acceleration vector $A(t) = \ddot{f}(t)$ equal to zero or collinear with the speed vector $V(t) = \dot{f}(t)$ except at some particular times where the derivative of the kernel is not continuous. Those kernels produce piecewise linear trajectories in the space S with kinks corresponding to discontinuities of the derivative. In such trajectories, only the points of discontinuity need to be considered as relevant points when computing the separation hyperplane. Compared to the case of L equidistant samples as used in sections 2.2 and 2.3 for which a parameter Δ requires to be fixed, this method provides a more simple and more effective set of points. Four such kernels already considered in Rotter and Diesmann (1999) are shown in **Figures 7A–D**.

We compared the generalization performance between the double exponential kernel used in section 2.3, a single exponential kernel considered in section 2.3, and a "RC kernel" with a time evolution described by the following equations.

$$k_{RC}(t) = \begin{cases} (1 - e^{-t/\tau}), & 0 \le t \le T_{\text{pulse}} \\ (1 - e^{-T_{\text{pulse}}/\tau})\, e^{-(t - T_{\text{pulse}})/\tau}, & t \ge T_{\text{pulse}} \end{cases}$$

T_{pulse} denotes the time where the double exponential function reaches its maximal value (i.e., $T_{\text{pulse}} = \tau\tau_r \log(\tau/\tau_r)/(\tau - \tau_r)$). For the three kernels, the time constant was set to $\tau = 13$ ms $= 1.3\,T$ where $T = 10$ ms is duration of spike patterns. The task was to separate 1 p^+ pattern from 5 p^- patterns. The number N of presynaptic neurons was equal to 10.

Figures 8A,B show that the single exponential kernel has lower performance, both with regards to FN and FP errors, than the bio-mimetic and the RC kernel, which have similar generalization error rate. Triangular, square, and single exponential kernels imply linear trajectories but are highly discontinuous, jumping instantaneously from 0 to 1 when a spike occurs. This causes severe problems in generalization task. The RC kernel, in contrast, has some advantages: each spike creates two kinks in the trajectory, the number of points to consider $\{f(t_{ij}, f(t_{ij} + T_{\text{pulse}}))\}$ is thus $N_k = 2\,N_s$ where N_s is the number of presynaptic spikes, it provides "smoother" trajectories and, therefore, better generalization performance compared to the three other kernels and it corresponds to a simple and well-known RC circuit that received at time t a current pulse of duration T_{pulse}.

While providing similar generalization results as a curved double exponential kernel, using the RC kernel with the SVM-PSP method allows much faster computation especially for sparse spike patterns.

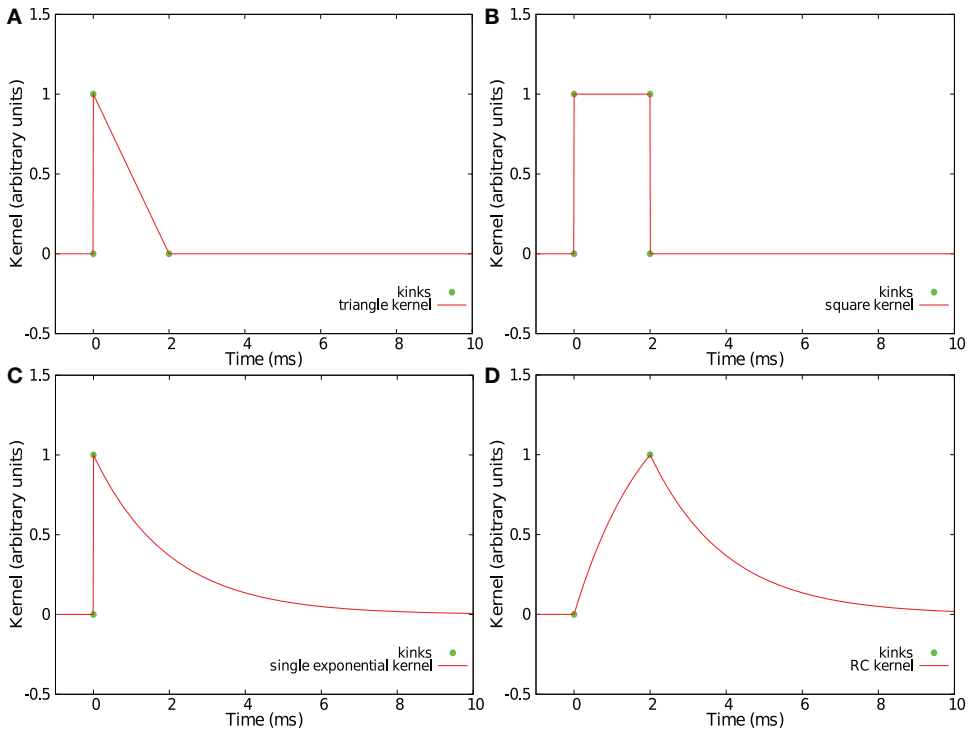

FIGURE 7 | Examples of kernels leading to piecewise linear trajectories in feature space. Shown are a triangular kernel **(A)**, a square kernel **(B)**, a single exponential kernel **(C)**, and a RC-kernel **(D)**.

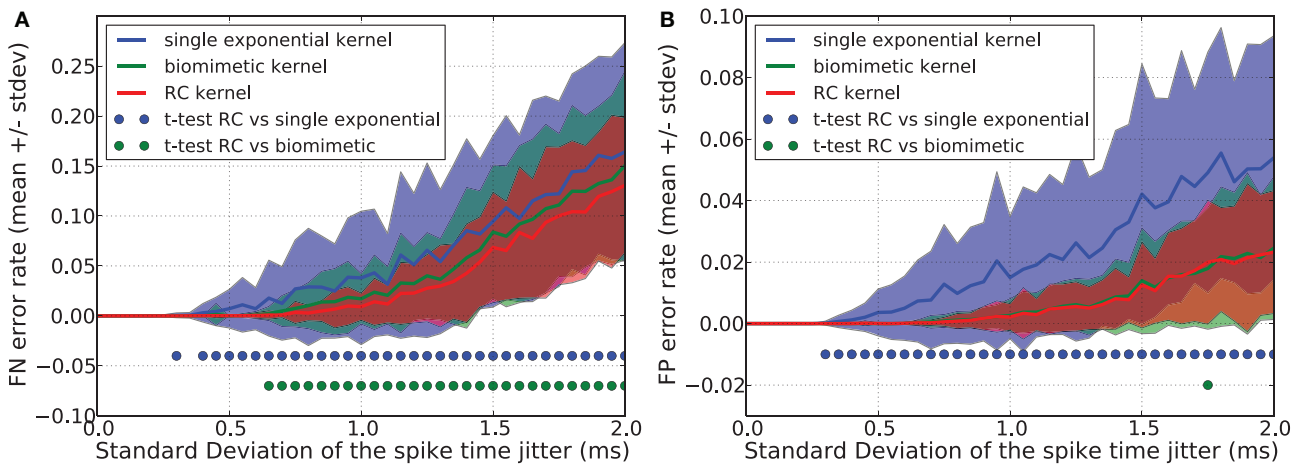

FIGURE 8 | Generalization performance measured as the number of misclassified noisy patterns over the total number of tested noisy patterns. Comparison between a single exponential kernel, a RC kernel and a bio-mimetic kernel. The task is to separate one target pattern from 5 background patterns. Mean value and standard deviation of false negative (FN) error **(A)** and false positive (FP) error **(B)**. Blue (resp. green) dots denote a significant difference of the generalization results (paired *t*-test, $p < 0.05$) obtained with the RC kernel and the single exponential kernel (resp. biomimetic kernel).

3. MATERIALS AND METHODS

3.1. SPIKE PATTERN GENERATION AND GENERALIZATION ERROR MEASURE

To generate spike patterns used in learning tasks, the time, t_i, of the spike of neuron $i \in \{1, \ldots, N\}$ in a pattern p, is given by $t_i = T_{\min} + (u(p, i) - 1)(T_{\max} - T_{\min})/(N - 1)$, where, for each p, $\{u(p, 1), u(p, 2), \ldots, u(p, N)\}$ is an independently chosen, random permutation of $\{1, 2, \ldots, N\}$, which indicates the order in which the neurons fire. $T_{\min}$ and $T_{\max}$ are the minimum and the maximum time of spike reception. We used $T_{\min} = 10$ ms

and $T_{max} = 20$ ms. $T = T_{max} - T_{min}$ denotes the spike pattern period. This is true only if no jitter is present. In the case of jitter, some spikes may come to lie outside of $[T_{min}, T_{max}]$. With this procedure, each spike pattern consist in N neuron firing one spike between T_{min} and T_{max} in a random order with a constant ISI.

To test generalization performance in sections 2.2, and 2.4, random jitter is introduced in each spike i of each learned spike pattern j. A new spike time t_{ij}^* is generated by adding a Gaussian noise Noise$(0, \sigma)$ to each spikes time t_{ij}. If $t_{ij}^* \leq 0$ ms or $t_{ij}^* > 30$ ms, a new noise value is pulled again.

Noisy p^{+*} and p^{-*} patterns created from each learned p^+ and p^- pattern are tested for classification. The simulation is repeated with a noise parameter σ that is varied between 0 and 2 ms.

FN error rate is the number of noisy p^{+*} patterns that have not been detected divided by the total number of noisy p^{+*} patterns. FP error rate is the number of noisy p^{-*} patterns that have been detected divided by the total number of noisy p^{-*} patterns.

In section 2.2, the statistical significance of the generalization ability of our method against the Tempotron and the voltage-margin Tempotron were tested with a paired t-test. In section 2.4, the same procedure was applied to test the generalization ability of a RC kernel against a bio-mimetic kernel, and a single-exponential kernel.

3.2. ORIGINAL DISCRETE TEMPOTRON

The Tempotron algorithm is based on the integrate-and-fire neuron model. The time evolution of the membrane potential V is described by the following equation

$$V(t) = V_{rest} + \sum_i w_i \sum_j k(t - t_{ij}) \tag{7}$$

in which t is the time, $V(t)$ is the membrane potential, w_i is a set of synaptic weights, t_{ij} is the time of the jth spike time of the ith presynaptic neuron, $k(t)$ is a postsynaptic kernel function and V_{rest} is the resting potential. N is the number of input neurons. The Tempotron is said to detect a spike pattern when its membrane potential exceeds a threshold value θ. For simplicity, we use $V_{rest} = 0$ and $\theta = 1$ in simulations.

The aim of the original Tempotron learning rule is to find an appropriate set of synaptic weights leading the membrane potential $V(t)$ above the threshold potential θ for spike patterns that should be detected (p^+), and to keep the membrane voltage below the threshold for background spike patterns that should not be detected (p^-). Again, using discrete time $t_l = \Delta\, l$, a pattern p is transformed into a set of voltage values $V_p = \{V(t_l)\}$. The stop condition of this learning rule is described by the following equations

$$\forall p \in P^+, \exists t_l : V(t_l) \geq \theta \tag{8}$$

$$\forall p \in P^-, \forall t_l : V(t_l) < \theta \tag{9}$$

For illustration, let us consider a task where two spike patterns, p^+ and p^-, have to be separated. A dot product between the two spike trains convolved with the kernel and those synaptic weights leads to two voltage traces $V^+(t)$ and $V^-(t)$ as described in Equation 7. If the maximum value of $V^-(t)$ occurs at $t_l = t_{max}^-$ and $V^-(t_{max}^-) = w_1 V_1(t_{max}^-) + w_2 V_2(t_{max}^-) + \cdots + w_N V_N(t_{max}^-) > \theta$, there is a misclassification. In this case, the Tempotron learning rule consists in the modification of all synaptic weights w_i by subtracting an amount of $\Delta_{w_i} = \lambda \sum_j k(t_{max} - t_{ij})$, λ being a learning coefficient. The same procedure is applied when $V^+(t_{max}^+) < \theta$, except that Δ_{w_i} is added to the synaptic weights. In this paper, synaptic weights are initialized to 0 and the λ parameter is equal to 0.1.

From a geometrical point of view as explained in section 2.1, fixing synaptic weights is equivalent to selecting a separation hyperplane according to equation $w_1 V_1 + w_2 V_2 + \cdots + w_N V_N - \theta = 0$, where θ is the bias of the hyperplane. Checking if there is a t_l for which $V^-(t_l) \geq \theta$ is equivalent to checking if a point $f^-(t_l)$ of F_p^- is above the current hyperplane, and modifying all synaptic weights w_i by an amount of Δ_{w_i} is equivalent to subtracting the vector $\lambda f(t_{max}^-)$ from the vector $w = (w_1, w_2, \ldots, w_N)$. Thus, the Tempotron and the SVM-PSP method share the same framework. The difference lies in the process of setting the hyperplane coefficients.

3.3. VOLTAGE-MARGIN TEMPOTRON

To maximize the voltage separation between p^+ and p^- patterns, the stop condition of the Tempotron algorithm can be modified by adding a voltage margin M_V as described in the following equations

$$\forall p \in P^+, \exists t_l : V(t_l) \geq \theta + M_V \tag{10}$$

$$\forall p \in P^-, \forall t_l : V(t_l) < \theta - M_V \tag{11}$$

The algorithm starts with a margin $M_V = 0$. When a separative set of synaptic weights has been found, the margin is increased by an amount of Δ_{M_V} that is fixed (we use $\Delta_{M_V} = 0.01$). The Tempotron learning rule is successively applied with an increasing margin. This process is repeated until a larger margin cannot be reached anymore. It is hard to decide whether a broader margin cannot be reached any more or if the algorithm requires more time to find it (except when the margin is equal to θ which is the highest margin that can be reached, the origin point $O = (0, 0, \ldots, 0)$ being in each p^- pattern). For this reason, the algorithm stops when the learning rule has been applied 100 times successively without finding a separation for a given margin.

Maximizing the measure $D_V = w_1 V_1 + w_2 V_2 + \cdots + w_N V_N - \theta$ is equivalent to maximizing $\|w\| D_S$ (see Equation 3). Let $\rho = \theta / \|w\|$ denotes the length of the orthogonal projection of the origin point O onto the separation hyperplane in the feature space S. From a geometric point of view, maximizing D_V amounts to finding a separation hyperplane that maximizes the separation of the point with the supplementary constraint of minimizing the distance ρ between the plane and the origin point.

3.4. GENETIC ALGORITHM

As explained in section 2.1, the classification task consists in finding at least one point $f^+(t_l)$ from each p^+ pattern that can be

linearly separated from all the points $f^-(t_l)$ of the p^- patterns. A genetic algorithm was used to test different combinations of $f^+(t_l)$ when several p^+ patterns have to be separated from background ones (p^-).

For this genetic algorithm, a genotype g is a tuple of cardinality $|G| = |P^+|$ in which each element (each gene) is one $f^+(t_l)$ belonging to each different p^+ pattern. A population is composed of M genotypes (we use $M = 8$). For each genotype g of the population, a separation hyperplane is calculated to try to separate all the points $f^-(t_l), p \in P^-$ from the points $f^+(t_l) \in g$. The associated fitness function is

$$\text{Fit}(G) = \min(\max(D_N(f^+(t_l))), -\max(D_N(f^-(t_l))))$$
$$\text{with } f^+(t_l) \in g \text{ and } f^-(t_l) \in P^-$$

Genotypes of the first generation are uniform random samples. After all genotypes of a population have been evaluated with the fitness function, the worse-performing half of the genotypes is discarded and replaced by randomly chosen new ones. The next quarter is used for reproduction, i.e., each pair of solutions is mixed by randomly interchanging one part of their genotypes to create two new solutions. Genotypes of the best quarter are used for mutation, i.e., randomly modifying the t_l of one gene ($f(t_l)$) by a uniform pull in the discrete range $[t_l - 5\Delta,\ t_l + 5\Delta]$. In section 2.2, to avoid a bias of any one algorithm in terms of learning cycles, we took the number of hyperplanes calculated by the voltage-margin Tempotron learning rule as the maximum number of genotypes tested by the genetic algorithm. A description of the algorithm is given here in list form:

1. Initialize the first generation by a random uniform sample
2. Evaluate the fitness function for all genotypes of this generation
3. Classify genotypes according to their fitness
4. Apply mutation and reproduction to the best 50% of all genotypes
5. Replace the worse-performing 50% of the genotypes by randomly chosen new ones
6. If the maximal number of genotypes to test is not reached, go to step 2.

4. DISCUSSION

In this paper, we have described a new bio-inspired method to classify spike patterns that provides better generalization results than the original Tempotron. Classification results of our method are both robust and easy to interpret in terms of neural networks. This can be interesting in electrophysiological data analysis where reliable spike burst classification is required. Moreover, because the algorithm is based on causal kernels, it can be adapted to online spike pattern classification.

Based on a geometric representation, we have obtained an analysis of the problem that reveals an intricate relation between PSP dynamics and classification capabilities of a Tempotron-like neuron. This new insights allow us to come up with a simple method to characterize computational tasks that different neuron types can solve. It might be interesting to further explore the relation between the statistics of input spike patterns (e.g., number of afferent neurons, oscillation frequencies in the afferent network) and the parameters of PSPs dynamics in the output neuron.

We introduced PSP kernel functions to support fast computations with the SVM-PSP method. In particular, the RC kernel provides good generalization performance. It might be used in electronic devices such as VLSI boards where the pulse-coded input is first transformed by a simple RC low-pass filter, while the output is compared to hyperplane equations at the start and at the end of each logical squared pulse (Mitra et al., 2009).

4.1. POSSIBLE IMPROVEMENTS OF THE SVM-PSP METHOD

In this paper, simulations have been performed with only one spike per neuron per pattern. However, the SVM-PSP method is not theoretically limited to such patterns. Receiving several spikes per neuron modifies the maximal values reached in each dimension of the feature space. The re-scaling for SVM computation and the distance $D_N(t_{\max})$ have to be changed accordingly. In section 2.2 we have shown that optimizing the D_N measure is better than M_V for generalization purposes. Alternative notions of distance could lead to even better generalization results. These two aspects need some additional study in the future.

Furthermore, we have shown that using a genetic algorithm leads to better generalization results than the learning rule originally described along with the Tempotron, and enhanced with an increased voltage margin. However, the comparison has here been done on an easy task (i.e., separate 2 p^+ pattern from 4 p^- pattern with one spike from 10 neurons). It is not clear though whether the genetic algorithm can find a good separation in more difficult tasks where more patterns have to be separated.

Tempotron-like algorithms have the advantage of using a gradient descent of a cost function whereas a genetic algorithm performs a stochastic search. A hybrid solution to solve the optimization problem would consist in running a Tempotron-like learning rule to find a separation hyperplane according to the equation $w_1 V_1 + w_2 V_2 + \cdots + w_N V_N - \theta = 0$. The times $t^+_{\max}$ of maximum voltages corresponding to all p^+ patterns are kept and used as seed points for the SVM-PSP method to obtain better separation coefficients (synaptic weights) for generalization.

The chosen hyper-parameters of the SVM (solver type, cost, and epsilon) provide satisfactory generalization results compared to the Tempotron and the voltage-margin Tempotron as shown in **Figure 2**. Fine-tuning them with a grid-based search would only improve the generalization results in our case. This kind of tuning is very-likely needed for more difficult classification tasks. We checked if those parameters modify the shape of curves of **Figure 4**. We used a grid-search with a cost parameter value from 10, 10000, 10000000 and an epsilon value from 0.01, 0.00001, 0.00000001 but we did not find any noticeable change in the shape (data not shown).

4.2. POSSIBLE IMPROVEMENTS OF THE TEMPOTRON METHOD

As previously mentioned, the training procedure of the original Tempotron algorithm stops without optimizing the separation hyperplane for generalization. To overcome this limitation,

a standard procedure would consist in generating a number of jittered training samples to learn the optimal weights. But this method entails some drawbacks. First, it is difficult to find the optimal amplitude of the noise injected to generate the training sample: using a too low noise level would not lead to a better separation hyperplane. Using a too high noise level could render the problem not linearly separable, which leads to bad classification results. Secondly, the multiplication of training patterns leads to an increased computational load. The SVM-PSP method avoids these problems by directly optimizing the margin between the points and the separation hyperplane. Using jittered training samples, the difference in the performance of the standard Tempotron and the voltage-margin Tempotron would become negligible only when the maximum voltage-margin is very small, close to compromising linear separability. In this case, the SVM-PSP method would theoretically also lead to the same separation hyperplane.

Changing the kernel from the curved double exponential kernel to a kernel that generates linear piece-wise trajectories in the phase space (such as the RC kernel) does not affect the general scheme of the discretized Tempotron learning rule. By using the RC kernel, the time discretization is simplified. Times that need to be computed are $V_p = \{V(t_{ij}), V(t_{ij} + T_{\text{pulse}})\}$. This can lead to a drastic reduction of computation time, especially in the case of sparse activity. The non-discretized version of the Tempotron also suggests a good method to decrease the number of relevant points by analytically resolving the times of maximum voltage between successive incoming spikes. It can be applied to double-exponential kernels, which are more similar to PSP in real nerve cells than "linear" kernels, such as the RC kernel. However, at each synaptic weight modification, and for each spike of each pattern, the non-dicretized version of the Tempotron requires the estimation of four parameters: the voltage, the synaptic current, the time of maximum voltage between two consecutive spikes, and the maximum voltage at this time. With the RC kernel for which kink times are fixed, at each synaptic weight modification, and for each spike of each pattern, only two numbers need to be determined: the voltage value at the two kink times.

One important question is whether the simple Tempotron-like neuron model is complex enough to also reflect the behavior of biological neurons. One of the most problematic differences is that biological synapses can not change their nature from excitatory to inhibitory, and *vice versa*. Another characteristic difference is that inhibitory synapses might have different kinetics than excitatory synapses (Karnup and Stelzer, 1999). Whereas with the SVM-PSP method it seems impossible to deal with these issues, one can partly resolve them by slightly modifying the Tempotron learning rules. Synaptic weights have to be initialized with the right sign, and the learning rule is modified preventing a synaptic weight to change its sign. Thus, it is possible to fix different PSP kernels for excitatory and inhibitory synapses.

ACKNOWLEDGMENTS

Funding by the German Ministry of Education and Research (Bernstein Focus Neurotechnology Freiburg/Tübingen, FKZ 01 GQ 0830) is gratefully acknowledged. We thank Moritz Deger, Volker Pernice, and Thomas Voegtlin for their critical comments on previous versions of the manuscript.

REFERENCES

Abeles, M., and Gerstein, G. L. (1988). Detecting spatiotemporal firing patterns among simultaneously recorded single neurons. *J. Neurophysiol.* 60, 909–924.

Ambard, M., Guo, B., Martinez, D., and Bermak, A. (2008). "A spiking neural network for gas discrimination using a tin oxide sensor array," in *4th IEEE International Symposium on Electronic Design, Test and Applications (DELTA)*, (Hong-Kong), 394–397.

Boahen, K. (2002). A retinomorphic chip with parallel pathways: encoding on, off, increasing, and decreasing visual signals. *J. Analog Integ. Circ. Signal Process.* 30, 121–135.

Bohte, S. M. (2004). The evidence for neural information processing with precise spike-times: a survey. *Trends Neurosci.* 3, 195–206.

Bohte, S. M., Kok, J. N., and Poutré, H. L. (2000). Spikeprop: backpropagation for networks of spiking neurons. *ESANN* 48, 17–37.

Brown, E. N., Kass, R. E., and Mitra, P. P. (2004). Multiple neural spike train data analysis: state-of-the-art and future challenges. *Nat. Neurosci.* 7, 456–461.

Chen, H. T., Ng, K. T., Bermak, A., Law, M. K., and Martinez, D. (2011). Spike latency coding in biologically inspired microelectronic nose. *IEEE Trans. Biomed. Circ. Syst.* 5, 160–168.

Christen, M., Kern, A., Nikitchenko, A., Steeb, W. H., and Stoop, R. (2004). Fast spike pattern detection using the correlation integral. *Phys. Rev.* 70, 011901.

Fan, R. E., Chang, K.-W., Hsieh, C. J., Wang, X. R., and Lin, C.-J. (2008). Liblinear: a library for large linear classification. *J. Mach. Learn. Res.* 8, 1871–1874.

Fries, P. (2009). Neuronal gamma-band synchronization as a fundamental process in cortical computation. *Annu. Rev. Neurosci.* 32, 209–224.

Grün, S., and Rotter, S. (2010). *Analysis of Parallel Spike Trains, Vol. 7.* New York, NY: Springer Series in Computational Neuroscience.

Gütig, R., and Sompolinsky, H. (2006). The tempotron: a neuron that learms spike timing-based decisions. *Nat. Neurosci.* 9, 420–428.

Houghton, C., and Sen, K. (2008). A new multi-neuron spike-train metric. *Neural Comput.* 20, 1495–1511.

Karnup, S., and Stelzer, A. (1999). Temporal overlap of excitatory and inhibitory afferent input in guinea-pig ca1 pyramidal cells. *J. Physiol.* 516, 485–504.

Kasiński, A., and Ponulak, F. (2006). Comparison of supervised learning methods for spike time coding in spiking neural network. *Int. J. Appl. Math. Comput. Sci.* 16, 101–113.

Kermany, E., Gal, A., Lyakhov, V., Meir, R., Marom, S., and Eytan, D. (2010). Tradeoffs and constraints on neural representation in networks of cortical neurons. *J. Neurosci.* 30, 9588–9596.

Maas, W. (1997). Network of spiking neurons: the third generation of neural network models. *Neural Netw.* 10, 1659–1671.

McKennoch, S., and Voegtlin, T. (2009). Spike-timing error backpropagation in theta neuron networks. *Neural Comput.* 21, 9–45.

Mitra, S., Fusi, S., and Indiveri, G. (2009). Real-time classification of complex patterns using spike-based learning in neuromorphic vlsi. *IEEE Trans. Biomed. Circ. Syst.* 3, 32–42.

Modha, D. S. (2011). Cognitive computing. *Commun. ACM* 54, 62–71.

Pan, L., Song, X., Xiang, G., Wong, A., Xing, W., and Cheng, J. (2009). First-spike rank order as a reliable indicator of burst initiation and its relation with early-to-fire neurons. *IEEE Trans. Biomed. Eng.* 56, 1673–1682.

Povysheva, N. V., Gonzalez-Burgos, G., Zaitsev, A. V., Kröner, S., Barrionuevo, G., Lewis, D. A., et al. (2006). Properties of excitatory synaptic responses in fast-spiking interneurons and pyramidal cells from monkey and rat prefrontal cortex. *Cereb. Cortex* 16, 541–552.

Rotter, S., and Diesmann, M. (1999). Exact digital simulation of time-invariant linear systems with applications to neuronal modeling. *Biol. Cybern.* 81, 381–402.

Rubin, R., Monasson, R., and Sompolinsky, H. (2010). Theory of spike timing-based neural classifiers. *Phys. Rev. Lett.* 105, 218102.

Schrauwen, B., and Campenhout, J. (2007). Linking non-binned spike train kernels to several existing spike train metrics. *Neurocomputing* 70, 1247–1253.

Urbanczik, R., and Senn, W. (2009). A gradient learning rule for the tempotron. *Neural Comput.* 21, 340–352.

van Rossum, M. C. W. (2001). A novel spike train distance. *Neural Comput.* 13, 751–763.

VanRullen, R., Guyonneau, R., and Thorpe, S. J. (2005). Spike times make sense. *Trends Neurosci.* 28, 1–4.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. New York, NY: Springer-Verlag.

Voegtlin, T. (2007). Temporal coding using the response properties of spiking neurons. *Adv. Neural Inf. Process. Syst.* 19, 1457–1464.

Conflict of Interest Statement and Publication Transfert: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. None of the material in this paper has been previously published or is under consideration for publication elsewhere.

Received: 15 February 2012; accepted: 02 November 2012; published online: 19 November 2012.

Citation: Ambard M and Rotter S (2012) Support vector machines for spike pattern classification with a leaky integrate-and-fire neuron. Front. Comput. Neurosci. ***6****:78. doi: 10.3389/fncom.2012.00078*

APPENDIX

A.1. NORMALIZED ORTHOGONAL DISTANCE BETWEEN A POINT AND THE SYNCHRONY VECTOR IN THE FEATURE SPACE

Let us consider the point $f(t) = (f_1(t), \ldots, f_N(t))$ in the feature space S. $f_s = (1, \ldots, 1)$ denotes the synchrony vector, $f'(t)$ is the orthogonal projection of the point $f(t)$ onto the vector f_s, and $l_s(t)$ is the length of the vector $f(t) - f'(t)$ normalized by $\|f_s\| = \sqrt{N}$. L_s is the maximum of $l_s(t)$ computed at all incoming spike times of one particular pattern. We found

$$
\begin{aligned}
f'(t) &= \langle f(t), f_s \rangle / \|f_s\|^2 f_s \\
&= (\bar{f}_i(t), \ldots, \bar{f}_i(t)) \\
l_s(t) &= \|f(t) - f'(t)\| / \|f_s\| \\
&= \sqrt{\frac{1}{N} \sum_i (f_i(t) - \bar{f}_i(t))^2} \\
&= \mathrm{std}(\{f_i(t)\}) \\
L_s &= \max(\mathrm{std}(\{f_i(t)\}))
\end{aligned}
$$