Cogn Comput (2014) 6:595–607
DOI 10.1007/s12559-014-9257-0

Novel Biologically Inspired Approaches to Extracting Online Information from Temporal Data

Zeeshan Khawar Malik · Amir Hussain · Jonathan Wu

Received: 8 September 2013 / Accepted: 17 March 2014 / Published online: 3 April 2014

Abstract In this paper, we aim to develop novel learning approaches for extracting invariant features from time series. Specifically, we implement an existing method of solving the generalized eigenproblem and use this to firstly implement the biologically inspired technique of slow feature analysis (SFA) originally developed by Wiskott and Sejnowski (Neural Comput 14:715–770, 2002) and a rival method derived earlier by Stone (Neural Comput 8(7):1463–1492, 1996). Secondly, we investigate preprocessing the data using echo state networks (ESNs) (Lukosevicius and Jaeger in Comput Sci Rev 3(3):127–149, 2009) and show that the combination of generalized eigensolver and ESN is very powerful as a more biologically plausible implementation of SFA. Thirdly, we also investigate the effect of higher-order derivatives as a smoothing constraint and show the overall smoothness in the output signal. We demonstrate the potential of our proposed techniques, benchmarked against state-of-the-art approaches, using datasets comprising artificial, MNIST digits and hand-written character trajectories.

Keywords Slow feature analysis · Echo state network · Generalized eigenvalue problem · Recurrent Neural Network · GenEigSfa · Stone's criterion · Higher-order changes

Z. K. Malik (✉) · A. Hussain · J. Wu
University of Stirling, Stirling, Scotland
e-mail: zkm@cs.stir.ac.uk

A. Hussain
e-mail: ahu@cs.stir.ac.uk

J. Wu
University of Windsor, Windsor, Canada
e-mail: jwu@uwindsor.ca

Introduction

Signals in nature tend to be more slowly changing than any noise or interference in their reception. This is also true of human depiction of nature—a video is not merely a random sequence of changing pixels but contains structures as the actors move continuously about the scene. Here, the question arises as to how we can consistently recognize an actor if it continuously changes in angle, eye position, distance, size and orientation. Learning invariances has always been a very important and interesting area of interdisciplinary research. There are many contributions in this area where novel biologically inspired learning approaches for extracting invariant features have been proposed.

In [9], the author has proposed a novel learning method which allows the network to learn transformation invariance by introducing a modified Hebbian learning approach between the simple and complex units which is a good way to implement positional invariance criteria but still the method is not able to extract invariance in a fast manner and in fact the method is rather slow and computationally expensive as well.

Similarly in [24], the author has proposed an invariant learning rule based on anti-Hebbian learning. The reason for using an anti-Hebbian learning rule instead of the Hebbian rule is to get an inverse effect on each neuron's output, thus extracting the least changeable features from the output. The only deficiency in this method is the high increase in computational complexity with the increase in size of the data. Another interesting unsupervised criteria for invariances were proposed by James Stone [26] in 1996, which is also one of the criteria which we investigate in this paper. Stone proposed a method of extraction of salient visual parameters using spatial temporal constraints.

The learning rule is based on the linear combination of Hebbian and anti-Hebbian weight learning, and the criteria are to learn salient visual parameters that changes very slowly by maximizing the long-term variance of each unit's output and simultaneously minimizing the short-term variance. Minimizing the short-term variance and maximizing the long-term variance simultaneously through a multilayer neural model is one more interesting criteria for extracting invariances, but we show later that this model does not consistently find invariant features.

Recently, Laurenz Wiskott has proposed the concept of slow feature analysis [33], a novel criteria for invariances, at the beginning of the twenty-first century. Slow feature analysis is an unsupervised learning technique which extracts more than one slowly varying features from the input dataset in a way that all the sets of features are decorrelated from each other but in some sense expresses information that is to some extent relevant to each other. Slow feature analysis is also another main area of our investigation in this paper. Slow feature analysis has been widely used in many areas of machine learning from pattern recognition [4] to reinforcement learning [18]. The slowest feature is a lowest eigenvector of the whitened covariance matrix. The optimized set of features extracted by SFA (slow feature analysis) reduces the search space for downstream goal-directed learning procedures [18]. Suppose a robot is placed in a search space, and it is sensing from an on-board camera. Reinforcement learning directly applied to the pixels of the image read by the robot will be computationally quite expensive due to the size of the search space but instead if first slow feature analysis technique is applied on the pixels of the image, then it can extract a small set of the variable's state and the robot can easily use this set of state variables to quickly develop useful control policies. In [4], the author has employed SFA (slow feature analysis) techniques for solving pattern recognition problem. The benefit demonstrated by the author in [4] showed that the extracted features of patterns belonging to the same class are similar because the slow feature analysis technique learns salient features of time series in a way invariant to frequent transformation. The other advantage of this technique is that the optimized set of features that needs to be considered for classification reduces the computational complexity as well. The author in this paper has expanded the original input by taking into consideration the square terms and the cubic terms while extracting slow features by solving the generalized eigenvalue problem. In [2], the authors extract slow features from an expanded time series data converted using reservoir computing and then employ an independent component analysis (ICA) approach for learning sparse coding on the SFA layer, primarily for robot localization applications. Similarly in [35], the authors apply slow feature analysis for human action detection. The cuboid of images related to different actions is initially read, and unsupervised slow feature extraction is performed on the cuboid of images. Following unsupervised extraction, the authors proposed and applied three novel supervised SFA learning techniques for classification of extracted features to detect human action, termed: Supervised SFA, Discriminative SFA and Spatial Discriminative SFA. The common goal of the three supervised learning techniques is to extract slow feature function of each category from the collected learned labeled cuboid. This approach is interesting but is computationally expensive as it involves many stages involved in recognizing a human action. However, the proposed SFA proved very useful in estimating the single underlying driving force with high accuracy on noisy input datasets. On the contrary in [34], the author has employed the SFA technique for estimation of a slowly varying single driving force in nonstationary time series.

More recently, a number of enhancements to the SFA approach have been proposed. For example, Kompella et al. [17] developed an incremental method for slow feature analysis: It is based on a combination of candid covariance-free incremental principal component analysis [31] and covariance-free incremental minor component analysis [21]. However, this algorithm proceeds in two stages: the first stage whitens the data and removes any lower-order principal components which are assumed to be noise before the second stage of performing the covariance-free minor component analysis. Thus, it cannot be truly called an on-line algorithm. Similarly in [28], the authors have deeply analyzed the working of SFA technique and interpret the algorithm as maximum-likelihood learning, by interpreting SFA's weights as recognition weights derived from the statistical inverse of a generative model. The authors derive a new soft version of SFA based on the probabilistic model free from any constraint on the output signal. In [5], the authors present an analytical comparison between SFA and second-order independent component analysis (ICA) by taking into account the time delay function in the two approaches. The authors have proved that in the case of the same time delay, both approaches are the same.

In this paper, we investigate an online technique for extracting structure from streams of sensory data: We use two criteria as previously mentioned, slow feature analysis [33] and that based on spatial temporal smoothness constraints, [26]. We do this using an existing incremental method for solving generalized eigenproblems. The rationale behind deriving a purely incremental version of the above mentioned techniques is inspired by the cognitive human vision system, which is not only able to learn both the fast and slow features in real-world scenes in a purely incremental manner, but can also simultaneously extract useful meaning from spatio-temporally varying data [8, 23]. According to one of the recent researches [16], it is shown that habits are also incrementally learned by human being through regular associations. Similarly, it is also proved that a slight incremental change in belief about intelligence in human being also influences their attention to information association with successful error correction [20]. Human brains have an excellent capacity to incrementally learn new skills and adapt to new environment [12]. This amazing incremental learning strength of the human brain in adapting all the surrounding drifts was the starting point for us to derive a novel purely incremental version of standard slow feature analysis technique.

We also investigate the use of reservoir computing, in particular echo state networks (ESNs), for providing projections of time series, each of which retains some memory of previous values of the time series. We give mathematical details of the ESN in section 'Echo State Networks.' The echo state network is one of the most popular types of recurrent neural network (RNN) proposed by Jaeger [15] in 2001 that is driven by a (one- or multidimensional) time signal. They are used to perform classification/regression on temporal data. Echo state networks have been used widely to solve various modeling and dynamic time series problems.

In [14], the authors have used echo state networks on a task of chaotic time series prediction. Similarly in [25], the authors have used an echo state network to model the prediction of speech signals in a classification experiment using isolated utterances of the English digits from 'zero' through 'nine.' Since echo state networks need no training from the input side to the reservoir, the learning is quite simple and less computationally expensive, and every digit can be learned in parallel by creating multiple output filters at readouts. In [22], the authors have used an echo state network to two examples of the generation of a dynamical model for a differential drive robot using supervised learning and secondly to the training of the respective motor controller. The movement of a robot in every direction is modeled by recurrent neural network (RNN), whereas an echo state network-based supervisor is employed, which dynamically trains a motor controller to make future predictions on the robot's movement.

In [6], the authors have used the dynamic echo state network to learn the temporal dependencies required to implement Q-learning while employing the concept of reinforcement Learning (RL). A dynamic echo state network-based learning architecture is presented that can represent temporarily dependent rewards for use in reinforcement learning (RL). Apart from the echo state network, effectiveness in the motor controller, dynamic time series prediction and memorizing musical sequences, the authors in [27] have used ESN on a natural language task and compared its performance with single recurrent network (SRN), and due to echo state network's short-term memory capability, it easily outperformed the other approach in prediction of the next word in complex sentences.

One of the authors of this paper [29, 30] has recently been involved in using ESNs for visualization of temporal data. Further, the time-varying characteristic in the data that is produced using echo state network (ESN) is also exploited by one of the authors in feed-forward neural network by developing a space variant neural network [7] based on autoregressive time-varying average process that performs simultaneous image restoration and blur identification.

Since ESNs are recurrent networks, they overcome some of the problems associated with existing recurrent networks, particularly the time-consuming process of learning the parameters of the network. We combine this with the method for solving eigenproblems to extract information from temporal data.

The remainder of this paper is organized as follows: Section 'An Incremental Method for Generalized Eigen problems' discusses the method, which we will use for solving eigenproblems. Section 'Echo State Networks' reviews echo state networks, and in section 'Slow Feature Analysis,' we discuss slow feature analysis and our new method, GenEigSfa; we discuss 2 possible versions of this method and provide simulations on artificial datasets. We also consider additional higher-order smoothing constraints in this section. In section 'Stone's Criterion,' we review James Stone's criterion and show that it can be realized using the generalized eigenproblem solver. Again, we implement the method on artificial data. The final sections apply the GenEigSfa method to MNIST digits and character trajectories datasets.

An Incremental Method for Generalized Eigenproblems

The generalized eigenproblem is the problem of finding the eigenvector **w** and eigenvalue λ which satisfy the equation

$$A\mathbf{w} = \lambda B\mathbf{w}, \tag{1}$$

where in this paper, A and B are two positive semidefinite matrices.

Zhang and Leung [36] show that one method of finding the maximum eigenvalue of the generalized eigenproblem is to iteratively use

$$\Delta\mathbf{w} = A\mathbf{w} - f(\mathbf{w})B\mathbf{w},$$
$$\mathbf{w} = \mathbf{w} + \eta\Delta\mathbf{w},$$

where η is a learning rate or step size. The function $f(\mathbf{w}) : R^n - \{0\} \rightarrow R$ satisfies

1. $f(\mathbf{w})$ is locally Lipschitz continuous
2. $\exists M_1 > M_2 > 0 : f(\mathbf{w}) > \lambda_1, \forall \mathbf{w} : \| \mathbf{w} \| \geq M_1$ and $f(\mathbf{w}) < \lambda_n, \forall \mathbf{w} : 0 < \| \mathbf{w} \| \leq M_2$
3. $\forall \mathbf{w} \in R^n - \{0\}, \exists N_1 > N_2 > 0 : f(\theta \mathbf{w}) > \lambda_1, \forall \theta : \theta \geq N_1$ and $f(\theta \mathbf{w}) < \lambda_n, \forall \theta : 0 \leq \theta \leq N_2$, and $f(\theta \mathbf{w})$, is a strictly monotonically increasing function of θ, in $[N_1, N_2]$.

where λ_1 is the greatest generalized eigenvalue and λ_n is the least eigenvalue.

Intuitively, what these criteria mean are that

1. The function is rather smooth.
2. It is always possible to find values of $\mathbf{w}_i, i = 1, 2$, large enough so that the functions of the weights exceed the greatest eigenvalue.
3. It is always possible to find values of $\mathbf{w}_i, i = 1, 2$, small enough so that the functions of the weights are smaller than the least eigenvalue.
4. For any particular value of $\mathbf{w}_i, i = 1, 2$, it is possible to multiply $\mathbf{w}_i, i = 1, 2$, by a scalar and apply the function to the result to get a value greater than the greatest eigenvalue.
5. Similarly, we can find another scalar so that, multiplying the $\mathbf{w}_i, i = 1, 2$, by this scalar and taking the function of the result gives us a value less than the smallest eigenvalue.
6. The function of this product is monotonically increasing between the scalars defined in 4 and 5.

This method was used in [10, 11] to perform extensions of canonical correlation analysis. In the next section, we will give a brief description of echo state network, which is used as a preprocessor before turning our attention to the two methods which comprise the heart of the paper.

Echo State Networks

Echo state networks (ESNs) [19] consist of three layers of ‘neurons’: An input layer which is connected with random and fixed weights to the next layer, which forms the reservoir. The reservoir is connected to the output neurons using weights which are trained using error descent. The diagram of the standard topology of echo state network is shown in Fig. 1. The neurons of the reservoir are connected to other neurons in the reservoir with a fixed, random, sparse matrix of weights. Typically, only about 10 % of the weights in the reservoir are nonzero. We emphasize that only the reservoir to output weights are trainable; the other sets of weights are fixed. It is this feature which gives the ESN the property of being easily and efficiently trained.

We first formalize the idea of reservoir. W_{in} denotes the weights from the N_u inputs $\mathbf{u}$ to the N_x reservoir units $\mathbf{x}$, W denotes the $N_x \times N_x$, reservoir weight matrix and W_{out} denotes the $(N_x + 1) \times N_y$, weight matrix linking the reservoir units to the output units, denoted $\mathbf{y}$. Typically, $N_x \gg N_u$, W_{in} is fully connected and fixed (i.e., the weights are nontrainable). In the standard echo state network, W is fixed but provides sparse connections: In this work, only 10 % of the weights in W are nonzero. W_{out} is a fully connected set of trainable weights (the ‘readout weights’). It is often stated that the W weights should be such that the spectral radius (its greatest eigenvalue) is less than 1 to ensure stability in the reservoir *when there is no input* (see (2)). However, a more useful heuristic for the more usual conditions (in which there is a nonzero input) is that there should be a playoff between the magnitude of the reservoir-reservoir weights, W, and those from the inputs, W_{in}: the larger the W is, the more the memory of previous values can be retained, but of course we cannot ignore the inputs entirely.

The network’s dynamics are governed by

$$\mathbf{x}(t) = f(W_{\text{in}}\mathbf{u}(t) + W\mathbf{x}(t-1)), \tag{2}$$

where $\mathbf{x}$ represents the generated activations, typically $f(.) = \tanh(.)$ and t is the time index. The feed-forward stage is given by

$$\mathbf{y} = W_{\text{out}}\mathbf{x}, \tag{3}$$

where $\mathbf{y}$ denotes the output of the reservoir. This is followed by a supervised learning of the output weights, W_{out}. If we are using online learning, a simple least mean square rule gives

$$W_{\text{out}} = W_{\text{out}} + \eta(\mathbf{y}_{\text{target}} - \mathbf{y})\mathbf{x}^T, \tag{4}$$

where η is a learning rate (step size) and $\mathbf{y}_{\text{target}}$ is the target output corresponding to the current input.

We will forego this output stage but use the reservoir activations as the data values before using the techniques of slow feature analysis to update the reservoir to output weights, W_{out}, so that the slow features of the time series can be identified.

Slow Feature Analysis

Wiskott and Sejnowski [33] developed a method of finding the filters of a dataset, which captured the most slowly changing features of the dataset. They argued that these features can be captured by a filter which reduced the magnitude of the rate of change of the data. Let $\mathbf{x}$ be an image vector. Then, the filter should try to reduce $E((\frac{d\mathbf{x}}{dt})^2)$ where $E(.)$ denotes the temporal expectation and the derivatives are with respect to time.

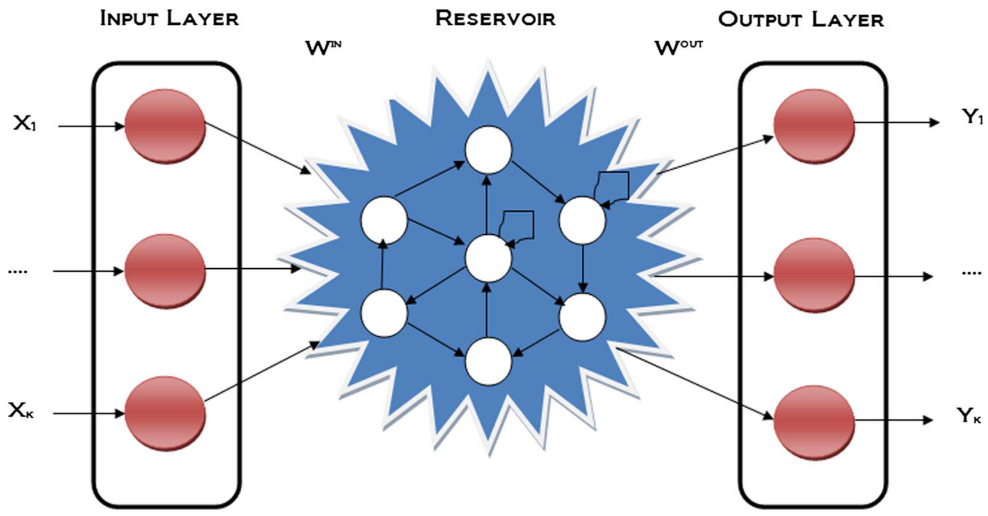

Fig. 1 Topology of echo state network

However, this alone is not enough: A filter which outputs a constant value would have $E(\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}) = 0$ but be totally uninformative. Thus, they argue that what is needed is a filter, **w**, which minimized this but also satisfied the conditions

$$E(\mathbf{w}^T\mathbf{x}) = 0,$$
$$E(\mathbf{w}^T\mathbf{x}\mathbf{x}^T\mathbf{w}) = 1,$$

with subsequent filters, if any, being orthonormal.

This was shown to be equivalent to the minimum of the generalized eigenvalue–eigenvector pair

$$E\left(\left(\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\right)\left(\frac{\mathrm{d}\mathbf{x}^T}{\mathrm{d}t}\right)\right)\mathbf{w} = \lambda E(\mathbf{x}\mathbf{x}^T)\mathbf{w}. \tag{5}$$

If we have zero mean data, we may consider the equations above as functions of the covariance matrices. The standard method [33] for solving (5) is to whiten the data so that $E(\mathbf{x}\mathbf{x}^T) = I$ and then perform a principal component analysis on the derivatives, which form the left-hand side of (5).

In the next section, we review a general method for solving such problems as slow feature analysis (SFA).

The GenEigSfa Method

We can apply the method of section 'An Incremental Method for Generalized Eigen problems' to the slow feature analysis (SFA) problem by finding the maximum eigenvalue of

$$E(\mathbf{x}\mathbf{x}^T)\mathbf{w} = \lambda' E\left(\left(\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\right)\left(\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\right)^T\right)\mathbf{w}, \tag{6}$$

where λ' is the inverse of λ in (5). This gives us an update of

$$\Delta\mathbf{w} = \Sigma_x\mathbf{w} - f(\mathbf{w})\Sigma_{\dot{x}}\mathbf{w}, \tag{7}$$

where $\Sigma_x = E(\mathbf{x}\mathbf{x}^T)$ and $\Sigma_{\dot{x}} = E\big((\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t})(\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t})^T\big)$.

In fact for a truly neural-based solution, inspired by the brain's incremental learning capability [32], i.e., updating the weights in an online mode, we can go further and replace the covariance matrices in (7) with the instantaneous values so that

$$\Delta\mathbf{w} = \mathbf{x}_i\mathbf{x}_i^T\mathbf{w} - f(\mathbf{w})\left(\frac{\mathrm{d}\mathbf{x}_\mathrm{i}}{\mathrm{d}t}\right)\left(\frac{\mathrm{d}\mathbf{x}_\mathrm{i}}{\mathrm{d}t}\right)^T\mathbf{w}. \tag{8}$$

We call this method the generalized eigenvalue slow feature analysis method (GenEigSfa).

Simulations

In our first simulation, we use

$$u(t) = \sin(t/33) + \cos(t/10 + \mu), \tag{9}$$

where $t = 1, \ldots, 2{,}000$ and $\mu = 3$ is an arbitrarily chosen phase term. We use a nonzero μ since without this term, each of the two signals may reinforce the other. We create a 20 dimensional input vector, and thus, we have 100 samples. Note that we did not use a sliding window since we wish to emulate real perception data and investigate a method, which is as biologically feasible as possible. (Although our subjective experience is of continuity, because of saccades and discrete spikes, our data are actually perceived in chunks.) To estimate $(\frac{\mathrm{d}\mathbf{x}_\mathrm{i}}{\mathrm{d}t})(\frac{\mathrm{d}\mathbf{x}_\mathrm{i}}{\mathrm{d}t})^T$ we simply used $(\mathbf{x}_i - \mathbf{x}_{i-1})(\mathbf{x}_i - \mathbf{x}_{i-1})^T$, a crude, but as we shall see, effective estimate. Thus, (8) becomes

$$\Delta\mathbf{w} = \mathbf{x}_i\mathbf{x}_i^T\mathbf{w} - f(\mathbf{w})(\mathbf{x}_i - \mathbf{x}_{i-1})(\mathbf{x}_i - \mathbf{x}_{i-1})^T\mathbf{w}. \tag{10}$$

In the top diagram of Fig. 2, we show the 2,000 samples from (9) and in the bottom diagram the filtered data using

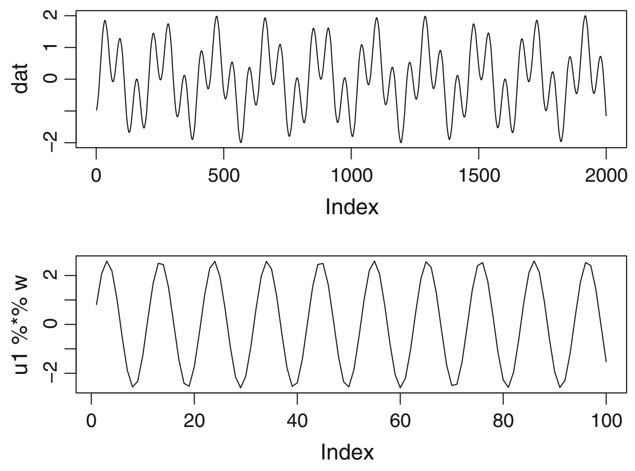

Fig. 2 *Top diagram* the data. *Bottom* the filtered data using GenEigSFA method

the trained values of $\mathbf{w}$ to filter the data. The low frequency has been clearly identified. We have tested a variety of functions for $f(\mathbf{w})$ and not found this to make significant differences to the results.

The linear method can only be applied when we have linear data ((9) is a linear combination of the two signals). We now consider a dataset also of 2,000 samples in which

$$u(t) = \sin(t/10)\cos(t/33) + \cos(t/10)\cos(t/33 + \mu). \tag{11}$$

We use the reservoir activations and their derivatives as the input to the SFA method, and results are shown in the bottom halves Figs. 3 and 4. We note from the top diagram (the data) that there is some beating apparent in these figures (giving around 3 periods), and it is this that the filtered data identifies. We do not always achieve this result: Sometimes one or other of the faster signals is identified ($\cos(t/10)$ or $\cos(t/33)$). Since the reservoir is randomly created each time, the filter found changes with each simulation.

The results above were created with a nonsparse reservoir. With a sparse reservoir, a sensible filter was found but the very slowest signal was not identified.

We have also experimented with moving windows so that

$$\mathbf{u}_1 = (u(1), u(2), \ldots, u(20))^T, \tag{12}$$

$$\mathbf{u}_2 = (u(2), u(3,), \ldots, u(21))^T, \quad \text{etc.} \tag{13}$$

but again a smooth filter was found similar to the filter from the sparse reservoir though sometimes a filter corresponding to a less slowly changing signal was found.

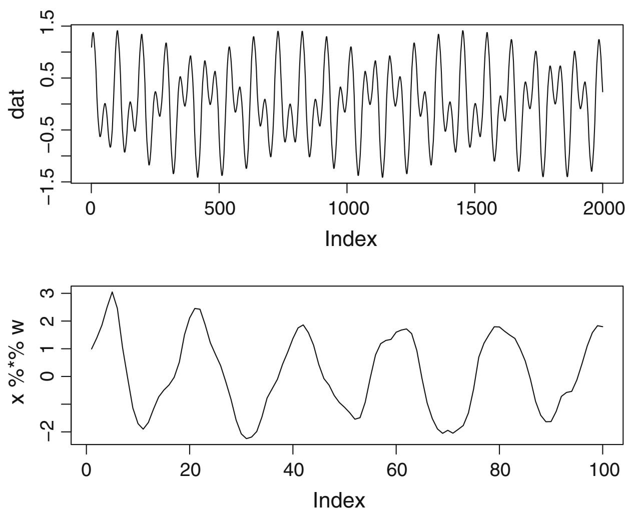

Fig. 3 *Top diagram* the data. *Bottom* the filtered data with reservoir as input to the GenEigSFA method

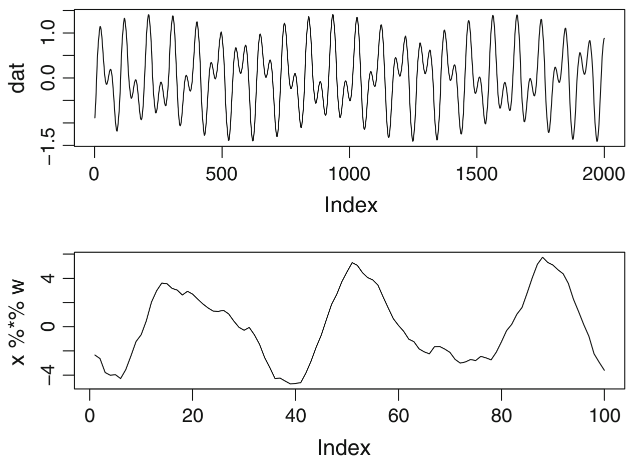

Fig. 4 *Top diagram*: the data. *Bottom*: the filtered data with reservoir as input to the GenEigSFA method

We note that not just any nonlinearity will do as preprocessing before the SFA stage: Using a radial basis function, we again failed to identify the slowest structure and indeed found little in the way of structured output at all.

Incorporating Higher-Order Changes

We may consider higher-order smoothness constraints by considering the effect of minimizing $E((\frac{d^n\mathbf{x}}{dt^n})^2)$, for $n = 2, 3, \ldots$. Then, this changes (7) to

$$E(\mathbf{x}\mathbf{x}^T)\mathbf{w} = \lambda' E\left(\left(\frac{d^2\mathbf{x}}{dt^2}\right)\left(\frac{d^2\mathbf{x}}{dt^2}\right)^T + \left(\frac{d\mathbf{x}}{dt}\right)\left(\frac{d\mathbf{x}}{dt}\right)^T\right)\mathbf{w}, \tag{14}$$

which gives us an update of

$$\Delta\mathbf{w} = \Sigma_x\mathbf{w} - f(\mathbf{w})[\Sigma_{\dot{x}} + \Sigma_{\ddot{x}}]\mathbf{w}, \tag{15}$$

where $\Sigma_x = E(\mathbf{x}\mathbf{x}^T)$, $\Sigma_{\dot{x}} = E\left(\left(\frac{d\mathbf{x}}{dt}\right)\left(\frac{d\mathbf{x}}{dt}\right)^T\right)$ and $\Sigma_{\ddot{x}} = E\left(\left(\frac{d^2\mathbf{x}}{dt^2}\right)\left(\frac{d^2\mathbf{x}}{dt^2}\right)^T\right)$.

As above, for a truly neural-based solution, i.e., updating the weights in online mode, we can go further and replace the covariance matrices in (15) with the instantaneous values so that

$$\Delta\mathbf{w} = \mathbf{x}_i\mathbf{x}_i^T\mathbf{w} - f(\mathbf{w})\left[\left(\frac{d\mathbf{x}_i}{dt}\right)\left(\frac{d\mathbf{x}_i}{dt}\right)^T + \left(\frac{d^2\mathbf{x}_i}{dt^2}\right)\left(\frac{d^2\mathbf{x}_i}{dt^2}\right)^T\right]\mathbf{w}. \tag{16}$$

Of course we can now consider putting more or less emphasis on different terms so that we have

$$\Delta\mathbf{w} = \mathbf{x}_i\mathbf{x}_i^T\mathbf{w} - f(\mathbf{w}) \left[\lambda\left(\frac{d\mathbf{x}_i}{dt}\right)\left(\frac{d\mathbf{x}_i}{dt}\right)^T + (1-\lambda)\left(\frac{d^2\mathbf{x}_i}{dt^2}\right)\left(\frac{d^2\mathbf{x}_i}{dt^2}\right)^T\right]\mathbf{w}, \tag{17}$$

for some $0 < \lambda < 1$.

We see in Fig. 5 that this method achieves a slightly better result than previously.

We also note that we can use a higher-order difference approximation to the covariance of the higher-order derivatives if we wish to maintain biological plausibility. Thus,

$$\frac{d^2\mathbf{x}_i}{dt^2} \approx (\mathbf{x}_{i+1} - \mathbf{x}_i) - (\mathbf{x}_i - \mathbf{x}_{i-1}). \tag{18}$$

Stone's Criterion

An early attempt [26] to extract invariances from visual data used a slightly different criterion: Stone argued that what was necessary was the extraction from visual signals of that part of a signal, which changed least. Of course an easy transformation is to simply map all input signals to a constant value, e.g., 0 but this presents an organism with no information about its environment at all. Therefore, we have to ensure that there is some variance in the output. This led Stone to suggest that the criterion which he wished to maximize was the ratio between the long-term variance, V and the short-term variance, U:

$$J = \log\frac{V}{U} = \log\frac{\sum_{t=1}^{T}(\tilde{z}_t - z_t)^2}{\sum_{t=1}^{T}(\bar{z}_t - z_t)^2}, \tag{19}$$

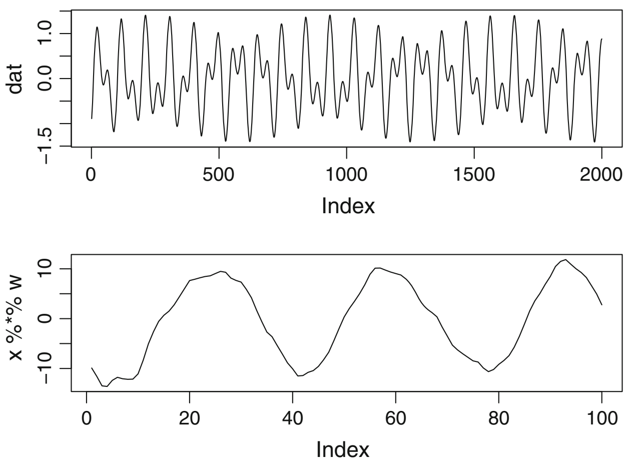

Fig. 5 Using second derivatives in GenEigSFA, we obtain slightly smoother results

where $\tilde{z}$ is an estimate of the long-term mean and $\bar{z}$ is an estimate of the local, short-term mean, both calculated by moving averages with appropriate smoothing parameters.

Thus, intuitively, Stone suggested that the output signal should contain as much variance as possible overall but as little variance as possible over the short term. Stone developed a learning method for updating a linear filtering parameter, which consisted of a mixture of Hebbian and anti-Hebbian learning.

We will use this criterion but with *exactly the same method as above*, i.e., the generalized eigenproblem solver of section 'An Incremental Method for Generalized Eigen problems.' We will pose the criterion as one of finding the generalized eigenvector of

$$\Sigma_L\mathbf{w} = \lambda\Sigma_S\mathbf{w}, \tag{20}$$

where Σ_L and Σ_S are the long-term and short-term estimates of the covariance matrices of the data, respectively. In practice, for our artificial data, we use $\Sigma_L = \Sigma$, the covariance matrix of the dataset and estimate Σ_S using a standard update rule such as

$$\Sigma_S = (1-\alpha)\Sigma_S + \alpha\mathbf{z}(t)\mathbf{z}^T(t), \tag{21}$$

where $0 < \alpha < 1$ is smoothing parameter.

Results

We begin with our artificial datasets. With the linear combination,

$$u(t) = \sin(t/33) + \cos(t/10 + \mu), \tag{22}$$

where $t = 1, \ldots, 1{,}000$ and $\mu = 3$ is an arbitrarily chosen phase term. We use the moving windows method so that

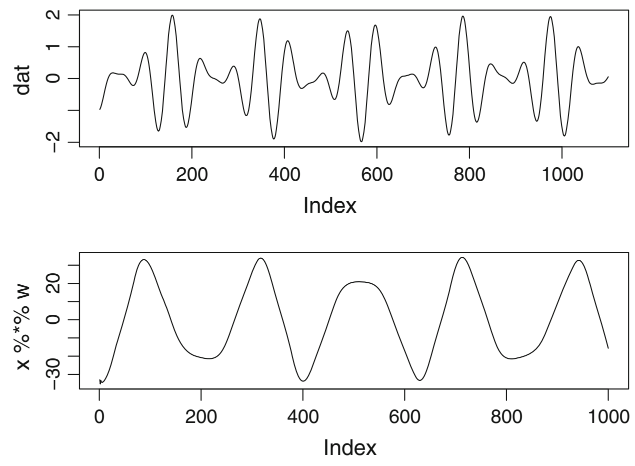

Fig. 6 Results from a reservoir with Stone's Method in which $N_x = 120$. Best results were when the size of the reservoir approximated the length of each data sample

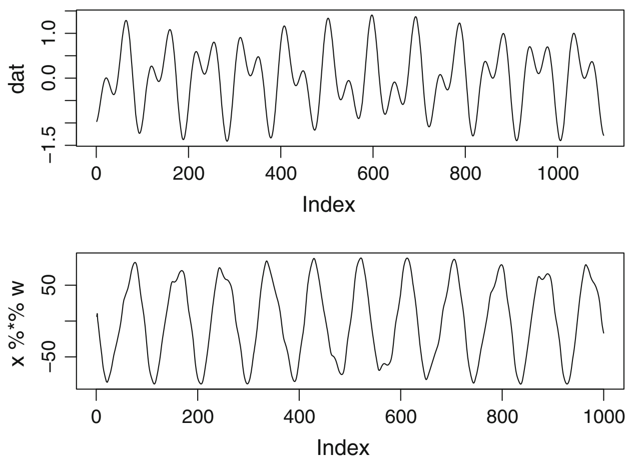

Fig. 7 Results from a reservoir with Stone's method in which $N_x = 120$. Best results were when the size of the reservoir approximated the length of each data sample

$$\mathbf{u}_1 = (u(1), u(2), \ldots, u(100))^T, \tag{23}$$

$$\mathbf{u}_2 = (u(2), u(3,), \ldots, u(101))^T. \tag{24}$$

etc.

Typical results are shown in Fig. 6 for which we used a reservoir of size $N_x = 120$ and 10000 iterations. The long-term covariance matrix was calculated just once but the short-term covariance matrix was updated using

$$\Sigma_S = \alpha\Sigma_s + (1 - \alpha)\mathbf{x}(t)(\mathbf{x}(t))^T, \tag{25}$$

with α set to values between 0.7 and 0.95. The results from Fig. 6 were achieved with $\alpha = 0.85$. We see that a slowly moving signal has been found but very much more crudely than with the SFA criterion. In general, we found Stone's criterion to produce less close approximations to the sinusoid signals than the SFA criterion.

Best results were achieved with a reservoir size approximately equal to the length of each data sample.

However, the Stone's reservoir method failed to find the slowest filter with the nonlinear combination data:

$$u(t) = \sin(t/10)\cos(t/33) + \cos(t/10)\cos(t/33 + \mu), \tag{26}$$

but it did find the second slowest, Fig. 7.

In previous sections, we have used our methods on artificial datasets generated by ourselves to illustrate the various methods. In the next sections, we illustrate the methods on two real datasets, which appear in the literature.

MNIST Dataset

We illustrate our group of methods which filter invariant features from the input data on MNIST handwritten digits dataset. The MNIST handwritten dataset consists of a standardized and freely available set of 70,000 handwritten digits. Each pattern consists of a handwritten digit of size 28 × 28 pixels. In the literature of slow feature analysis (SFA) [4, 13], this handwritten digit dataset is mostly used for pattern recognition, but here we are using this dataset to show how effective our algorithms are to identify the change. We read the digits, sort them from 0 to 9 into ascending order and consider 1,000 each. The reason for considering 1,000 digits of same type from each class is to form a sequential time series comprising similar patterns that belongs to the same class. We first incrementally learnt all the 0's in the time series, with an equal time interval assigned to the learning of each digit. The data of all 0's were then projected on to the filter that was obtained after learning 0's, in order to find the output. In order to identify the change, we also project the output of all 1's on to the same filter (obtained after learning 0's), so that we can compute the difference very easily by first projecting 0's on their own filter and then projecting 1's on the 0's filter, and so we can observe the differences in the respective outputs. To ensure that the slow features of each digit were learned, we presented each of the 1,000 samples of each digit 10 time to the network. We have done this for all the digits by first calculating output using their own filter and then using the same filter for calculating the output of next digit. The learning rate is taken as 0.00001, and the iteration for learning is 10 epochs of 1,000 samples.

In Figs. 8, 9, 10, 11 and 12, we show the results of digit 1 and 2. The x-axis in the figures represent the sample number, and the y-axis represents the actual output y.

We firstly in Fig. 8 show the results by using the standard slow feature analysis algorithm [33]. The change is identified by showing a difference in amplitude of the slow features coming from 2's according to filter of 1's then 1's on its own filter.

In Fig. 9, we show the results of digit 1 and 2 by using our GenEigSfa method. The slow features coming from 2's according to filter of 1's are different in amplitude from the slow features of 1's on 1's filter. This change in amplitude is clearly identifying the change in digit which our incremental algorithm is identifying while learning in sequence from 0 to 9.

We have tested our incremental algorithm on the activations produced by echo state network as well. The size of reservoir is 100, and the other parameters are the same as above. The input weights and weights of the reservoir are initialized randomly between 0 and 1. In Fig. 10, we can see that by using reservoir's activations the difference has become more pronounced, and the projections of digits on their own filter are quite different from the projections of digits on another digit's filter. Clearly, the combination of reservoir and our incremental slow feature analysis is very powerful.

The higher-order derivatives have also been tested with our incremental algorithm, and the results are shown in Fig. 11. The value of λ is taken as 0.5. The increase or decrease in the value of lambda is not making much difference in the amplitude of the output signal. The role of this smoothness constraint is to add smoothness in the output feature, which is not shown very clearly in our case but still the change in moving from one digit to the other is very prominent.

Next, we tested the alternative method which is based on Stone's approach. In Stone's method, the value of α is taken as 0.5. The learning rate and the number of iterations are unchanged. Results are shown in Fig. 11. Here, the change is not really prominent as compared to the other method where the change in amplitude is more clearer.

Even with reservoirs, the results with Stone's method are not really convincing as compared to the combination of reservoirs and GenEigSFA. The minimum and maximum values of the output for digit '1' on its own filter and digit '2' on 1's filter for all the techniques are shown in Table 1.

The experiment is further explained in Table 1. The main reason of creating Table 1 is to show the dissimilarity between the output of 1s projected on 1s filter and the output of 2s projected on 1s filter by showing their range of output and calculating the correlation coefficient between the output of both the signals. It is clearly shown in Table 1 that the correlation between the output of both the signals using GenEigSfa method with reservoir is lesser as compared to the other methods. This shows that GenEigSfa method with reservoir identify the change a bit more clearly as compared to the other methods where the change is lesser.

Character Trajectories

Next, we used the character trajectories dataset [3], which consists of 2,858 character samples. The categories of characters are from the letters a–z. Each character can have a different number of pixels, which is always between 174 and 205 in length with the standard 3-dimensional data giving x, y and z coordinates. We have considered first 1,000 characters as our training dataset and the remaining characters we have reserved as our test dataset. We first filter 20 characters from each category out of 1,000 characters. The rationale for extracting 20 characters of same type is to create a small sequential time series consisting of patterns belonging to same class that can be learned

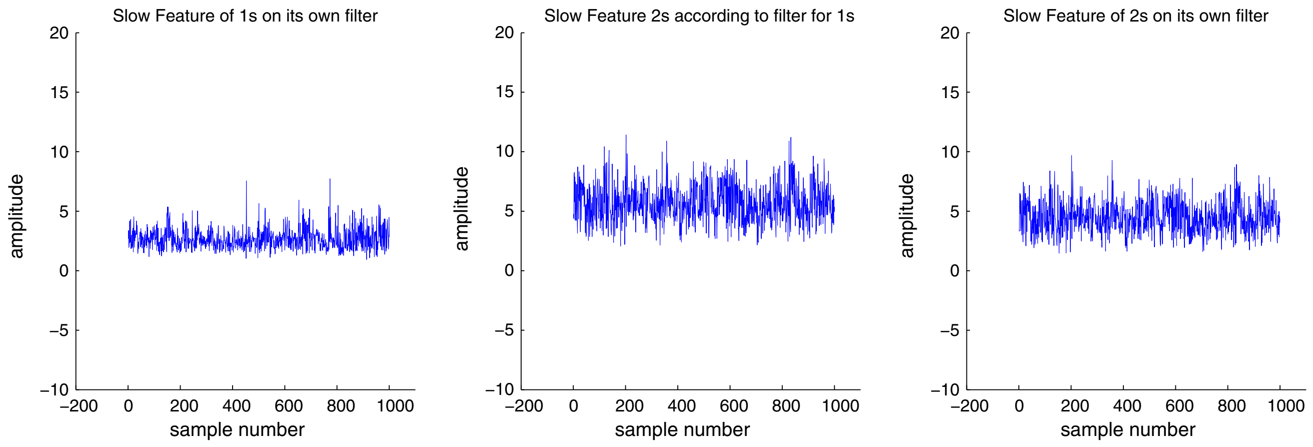

Fig. 8 Standard SFA [33]. *Left* Slow features of 1's on its own filter. *Middle* Slow features of 2's according to 1's filter. Slow features of 2's on its own filter

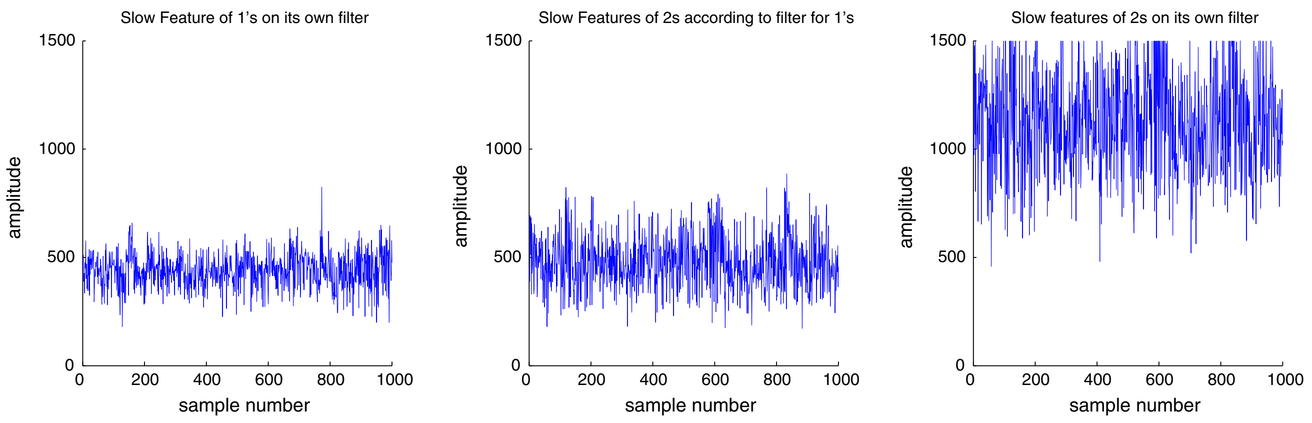

Fig. 9 Incremental SFA *Left*: Slow features of 1's on its own filter. *Middle*: Slow features of 2's according to 1's filter. Slow features of 2's on its own filter

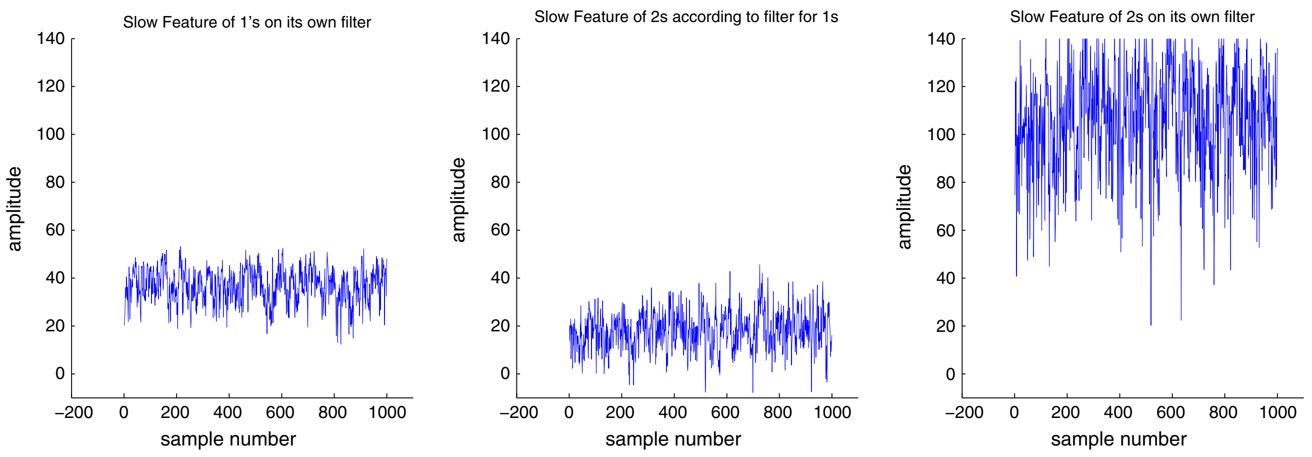

Fig. 10 Incremental SFA with Reservoir. *Left* Slow features of 1's on its own filter. *Middle* Slow features of 2's according to 1's filter. Slow features of 2's on its own filter

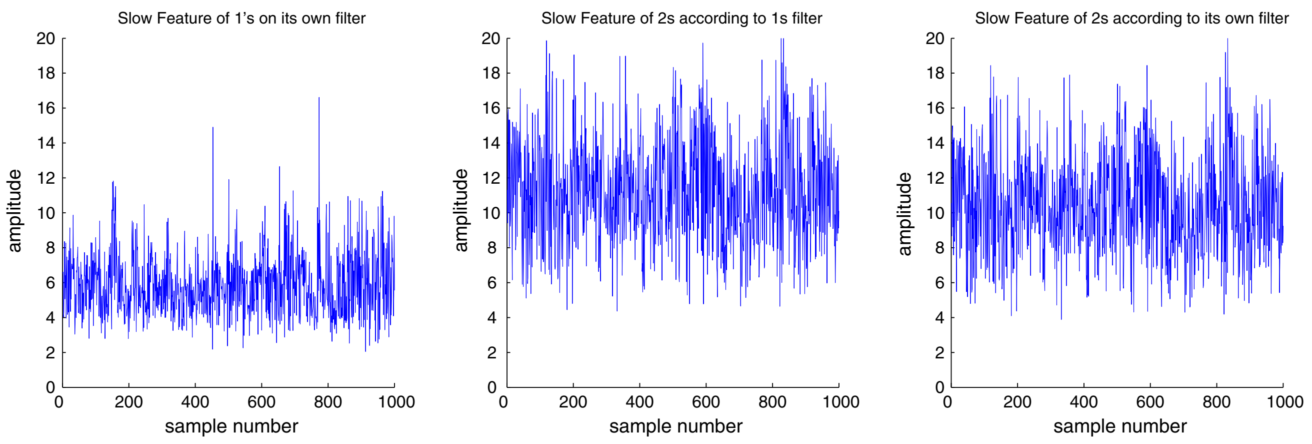

Fig. 11 Incremental SFA with Smoothness Constraint: Slow features of 1's on its own filter. *Middle* Slow features of 2's according to 1's filter. Slow features of 2's on its own filter

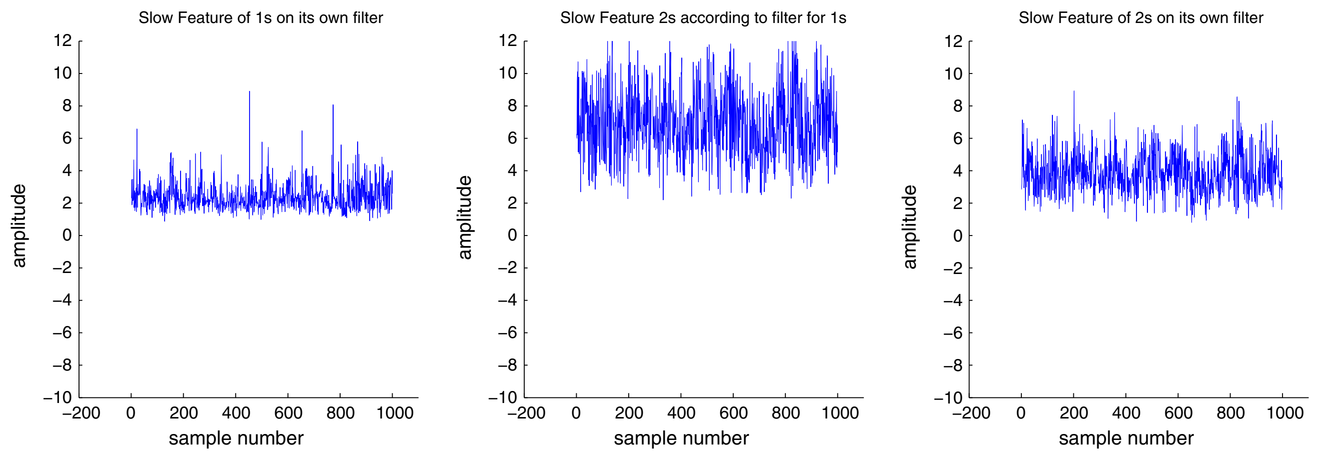

Fig. 12 Stone's Method: Slow features of 1's on its own filter. *Middle* Slow features of 2's according to 1's filter. Slow features of 2's on its own filter

Table 1 Comparison of magnitude of output signals

	Minimum	Maximum	CorrCoeff (y1, y2)
Standard slow feature analysis algorithm			
Range of output of 1s projected on 1s filter (y1)	0.9411	7.7330	0.9807
Range of output of 2s projected on 1s filter (y2)	2.0555	11.4144	
Incremental Learning (GenEigSfa)			
Range of output of 1s projected on 1s filter (y1)	180.1291	820.7507	0.9890
Range of output of 2s projected on 1s filter (y2)	172.282	831.8435	
Incremental learning (GenEigSfa) with reservoir			
Range of output of 1s projected on 1s filter (y1)	12.3146	53.1756	0.9479
Range of output of 2s projected on 1s filter (y2)	−7.8218	45.5810	
Incremental learning (GenEigSfa) with higher-order derivatives			
Range of output of 1s projected on 1s filter (y1)	2.0508	16.6154	0.9824
Range of output of 2s projected on 1s filter (y2)	4.3643	21.9050	
Incremental learning (GenEigSfa) using stone's criterion			
Range of output of 1s projected on 1s filter (y1)	0.8779	8.9089	0.9773
Range of output of 2s projected on 1s filter (y2)	2.1950	13.8449	

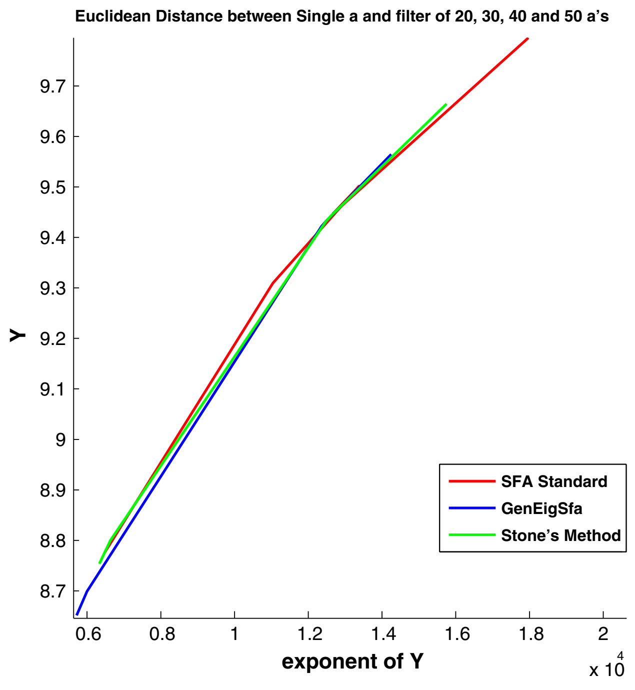

Fig. 13 Comparative euclidean distance graph

together in a sequential manner dedicating equal time interval to each character for learning and extracting a cumulative average filter. Out of each group of 20 characters of the same type, we choose the first group of a's and extract the average filter for all the a's. The same procedure has been done for all the other groups. After extracting the average filter from each group, we now tested the accuracy of our proposed methods. We choose a random 'a,' 'b,' 'c,' 'd' and 'e' character from the test dataset and project these characters onto the average filter of a's only. Meanwhile, a single 'a' is selected randomly from the test dataset, and the slow features of this character is extracted. We calculate the euclidean distance between the single character output and the projected a, b, c, d and e on the average filter of 20 a's, and the result is given in Table 2 shown in the appendix.

Discussion

As it can be seen from Table 1, our method has performed better compared to the standard SFA method because the euclidean distance between the output of 20 a's and the single 'a' by our method is less than that of the standard SFA method. The distance is shown graphically in an exponential way in Fig. 13 as well where *x*-axis represents the exponent of euclidean distance *Y*, and *y*-axis represents the actual euclidean distance *Y*. The blue line shows the distance of our GenEigSfa method, which is lesser as compared to the other methods and hence shows the lesser distance between the output produced by the filter of 20 a's and the output of single a projected on the filter of 20 a's. All the techniques were found to be successful in recognizing the 'a' character because the distance of output of character 'a' is less than other characters, which means that the slow feature of a is much closer to the average output of a's than the projection of other characters on the average filter of 20 a's. We have also tested the euclidean distance between the output of single 'a' and the output of 10, 20, 30, 40 and 50 a's and still found our results better than the standard SFA method highlighted in Table 2 shown in the appendix.

Conclusion

We have reviewed an incremental method for solving generalized eigenproblems and applied it to develop two novel techniques for extraction of information from temporal data. Specifically, we first proposed a purely incremental version of slow feature analysis (GenEigSfa), and secondly, we introduced an incremental version based on Stone's criterion both for the purpose of extracting invariant features from the data. Both the approaches were tested on artificial and real datasets. A new smoothing criterion using higher-order derivatives was also proposed and tested. We have also shown that it is often better to preprocess the inputs to the two information extraction techniques by using the outputs of an echo state network as input to these techniques.

We have shown that the SFA criterion is more powerful than Stone's criterion. Unfortunately, an existing technique in the literature is already called incremental slow feature analysis; however, our new technique is truly an incremental implementation of the solution of the slow feature analysis (SFA) criterion. Since our technique is based on the solution of a generalized eigenproblem, we call it GenEigSfa.

We have also shown using both simulated and real datasets that the combination of reservoir and SFA is very effective in automatically identifying classes of digits in image data. Current work is aimed at adapting the above method with different structures of reservoir and on developing an incremental slow feature analysis-based bi-clustering algorithm benchmarked against state-of-the-art bi-clustering approaches [1].

Acknowledgments The first author is grateful to Professor Colin Fyfe, formerly with the University of The West of Scotland, for his insightful suggestions, which helped improve the writing of this paper.

Appendix

See Table 2.

Table 2 Euclidean distance matrix between a single 'a' and the projected a, b, c, d and e on average filter for 20 a's

No. of a's	Single character	SFA standard	GenEigSfa	Stone's method
10	a	**9.8011**	**9.5646**	**9.6646**
10	b	12.7465	12.8941	12.5942
10	c	14.4252	14.5521	14.3502
10	d	27.2787	27.8709	27.0879
10	e	21.7497	21.6603	21.6602
20	a	**9.4469**	**9.4322**	**9.4422**
20	b	12.2026	12.4475	11.4916
20	c	14.0197	14.6512	13.6831
20	d	21.1748	26.1381	26.1505
20	e	21.8839	22.3803	21.3733
30	a	**9.5017**	**9.4221**	**9.4318**
30	b	11.4595	12.0080	10.0089
30	c	13.5355	12.7503	12.7511
30	d	26.7195	27.5374	24.5378
30	e	21.8156	22.7354	20.7354
40	a	**9.3095**	**8.6990**	**8.6998**
40	b	10.4456	11.6419	8.6429
40	c	13.0647	11.7851	11.7863
40	d	24.7328	24.3852	21.3862
40	e	20.7265	21.0699	19.0701
50	a	**8.7751**	**8.6513**	**8.6537**
50	b	9.6769	9.1272	8.1271
50	c	12.6343	12.0943	11.0930
50	d	23.2484	28.6332	18.6352
50	e	19.9421	20.7678	17.6574

References

1. Abdullah A, Hussain A. A new biclustering technique based on crossing minimization. Neurocomputing. 2006;69(16):1882–96.
2. Antonelo E, Schrauwen B. Learning slow features with reservoir computing for biologically-inspired robot localization. Neural Netw. 2012;25:178–90.

3. Bache K, Lichman M. UCI machine learning repository. Irvine, CA: University of California, School of Information and Computer Science; 2013. http://archive.ics.uci.edu/ml.
4. Berkes P. Pattern recognition with slow feature analysis. Cognitive Sciences EPrint Archive (CogPrints) 4104, 2005.
5. Blaschke T, Berkes P, Wiskott L. What is the relation between slow feature analysis and independent component analysis? Neural Comput. 2006;18(10):2495–508.
6. Bush K, Anderson C. Modeling reward functions for incomplete state representations via echo state network. In: Neural Networks, 2005. Proceedings. 2005 IEEE international joint conference on IJCNN'05, Vol. 5. IEEE.
7. Cheema TA, Qureshi IM, Hussain A. Blind image deconvolution using space-variant neural network approach. Electron Lett. 2005;41(6):308–09.
8. Ding Y, Song Y, Fan S, Qu Z, Chen L. Specificity and generalization of visual perceptual learning in humans: an event-related potential study. Neuroreport. 2003;14(4):587–90.
9. Földiák P. Learning invariance from transformation sequences. Neural Comput. 1991;3(2):194–200.
10. Gou Z, Fyfe C. A canonical correlation neural network for multicollinearity and functional data. Neural Netw. 2004;17(2): 285–93.
11. Gou Z, Fyfe C. A family of networks which perform canonical correlation analysis. Int J Knowl-Based Intell Eng Syst. 2001; 5(2):76–82.
12. Green CS, Bavelier D. Exercising your brain: a review of human brain plasticity and training-induced learning. Psychol Aging. 2008;23(4):692.
13. Huang Y, Zhao J, Tian M, Zou Q, Luo S. Slow feature discriminant analysis and its application on handwritten digit recognition. In: Neural Networks, 2009. International joint conference on IJCNN 2009. IEEE pp. 1294–7.
14. Jaeger H, Haas H. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science. 2004;304(5667):78–80.
15. Jaeger H. Short term memory in echo state networks. GMD-Forschungszentrum Informationstechnik. 2001.
16. Knowlton BJ, Mangels JA, Squire LR. A neostriatal habit learning system in humans. Science. 1996;273(5280):1399–402.
17. Kompella VR, Matthew L, Schmidhuber J. Incremental slow feature analysis: adaptive low-complexity slow feature updating from high-dimensional input streams. Neural Comput. 2012; 24(11):2994–3024.
18. Legenstein R, Wilbert N, Wiskott L. Reinforcement learning on slow features of high-dimensional input streams. PLoS Comput Biol. 2010;6(8):e1000894.
19. LukošEvičIus M, Jaeger H. Reservoir computing approaches to recurrent neural network training. Comput Sci Rev. 2009;3(3): 127–49.
20. Mangels JA, Butterfield B, Lamb J, Good C, Dweck CS. Why do beliefs about intelligence influence learning success? A social cognitive neuroscience model. Soc Cogn Affect Neurosci. 2006; 1(2):75–86.
21. Peng D, Yi Z, Luo W. Convergence analysis of a simple minor component analysis algorithm. Neural Netw. 2007;20(7):842–50.
22. Plöger PG, Arghir A, Gunther T, Hosseiny R. Echo state networks for mobile robot modeling and control. In: RoboCup 2003: Robert Soccer World Cup V11. Springer Berlin Heidelberg, 2004; p. 157–68.
23. Qu Z, Song Y, Ding Y. ERP evidence for distinct mechanisms of fast and slow visual perceptual learning. Neuropsychologia. 2010;48(6):1869–74.
24. Schraudolph NN, Sejnowski TJ. Competitive anti-hebbian learning of invariants. In: NIPS. Vol. 4. 1991.
25. Skowronski MD, Harris JG. Minimum mean squared error time series classification using an echo state network prediction model. In: Circuits and Systems, 2006. Proceedings. 2006 IEEE International Symposium on ISCAS 2006. IEEE.
26. Stone JV. Learning perceptually salient visual parameters using spatiotemporal smoothness constraints. Neural Comput. 1996;8(7):1463–92.
27. Tong MH, Bickett AD, Christiansen EM, Cottrell GW. Learning grammatical structure with echo state networks. Neural Netw. 2007;20(3):424–32.
28. Turner R, Sahani M. A maximum-likelihood interpretation for slow feature analysis. Neural Comput. 2007;19(4):1022–38.
29. Wang TD, Fyfe C. Visualising temporal data using reservoir computing. J Inf Sci Eng. 2013;29(4):695–709.
30. Wang TD, Wu X, Fyfe C. Factors important for good visualisation of time series. Int J Comput Sci Eng. (in press).
31. Weng J, Zhang Y, Hwang W. Candid covariance-free incremental principal component analysis. Pattern analysis and machine intelligence, IEEE Trans. 2003;25(8):1034–40.
32. Werbos PJ. Intelligence in the brain: a theory of how it works and how to build it. Neural Netw. 2009;22(3):200–12.
33. Wiskott L, Sejnowski TJ. Slow feature analysis: unsupervised learning of invariances. Neural Comput. 2002;14(4):715–70.
34. Wiskott L. Estimating driving forces of nonstationary time series with slow feature analysis; 2003. arXiv preprint cond-mat/0312317.
35. Zhang Z, Zhao M, Chow TW. Binary- and multi-class group sparse canonical correlation analysis for feature extraction and classification. Knowl Data Eng, IEEE Trans. 2013;25(10): 2192–205.
36. Zhang Q, Leung YW. A class of learning algorithms for principal component analysis and minor component analysis. Neural Netw, IEEE Trans. 2000;11(2):529–33.