Cogn Comput
DOI 10.1007/s12559-016-9439-z

Semi-supervised Echo State Networks for Audio Classification

Simone Scardapane[1] · **Aurelio Uncini**[1]

Received: 9 September 2016 / Accepted: 5 November 2016

Abstract Echo state networks (ESNs), belonging to the wider family of reservoir computing methods, are a powerful tool for the analysis of dynamic data. In an ESN, the input signal is fed to a fixed (possibly large) pool of interconnected neurons, whose state is then read by an adaptable layer to provide the output. This last layer is generally trained via a regularized linear least-squares procedure. In this paper, we consider the more complex problem of training an ESN for classification problems in a *semi-supervised* setting, wherein only a part of the input sequences are effectively labeled with the desired response. To solve the problem, we combine the standard ESN with a semi-supervised support vector machine (S^3VM) for training its adaptable connections. Additionally, we propose a novel algorithm for solving the resulting non-convex optimization problem, hinging on a series of successive approximations of the original problem. The resulting procedure is highly customizable and also admits a principled way of parallelizing training over multiple processors/computers. An extensive set of experimental evaluations on audio classification tasks supports the presented semi-supervised ESN as a practical tool for dynamic problems requiring the analysis of partially labeled data.

✉ Simone Scardapane
simone.scardapane@uniroma1.it

[1] Department of Information Engineering, Electronics and Telecommunications (DIET), "Sapienza" University of Rome, Via Eudossiana 18, 00184, Rome, Italy

Keywords Echo state network · Reservoir computing · Semi-supervised learning · Audio classification · Non-convex optimization · Parallel computing

Introduction

Recurrent neural networks (RNNs) are a powerful tool for the analysis of dynamic data, thanks to their capacity of modeling highly non-linear relationships and temporal dynamics [7, 54]. Classical RNN training, in which all the weights of the RNN are updated via an iterative optimization strategy, can however suffer from the problems of the exploding and vanishing gradients, wherein the norm of the weights' update can exponentially go to zero (or diverge) when it is back-propagated through time [36]. This, in turn, requires the use of expensive or ad hoc solutions, such as gated neuronal units [20], highly customized quasi-Newton optimization algorithms [32], and gradient clipping strategies [36, 53].

For problems requiring only a limited amount of memory, an efficient alternative to optimizing the full RNN is the echo state network (ESN), proposed in 2001 by H. Jaeger [21–23]. In an ESN, the recurrent portion of the network (called *reservoir*) is fixed in advance, generally by pseudo-random assignment of its weights. Then, a *readout* layer of output connections is adapted based on the training data. Owing to the topology of the network, this training problem is reduced to a standard linear regression routine (as shown in the following section), which can be solved efficiently even on very large datasets [43]. Despite its inherent simplification, the ESN can achieve state-of-the-art results

Published online: 21 November 2016

in many domains, including short-term forecasting [6, 27], chaotic series prediction [25, 45], restricted acoustic modeling [49], grammar learning [47], dynamic modeling [13], and many others [33]. Belonging to the broader family of "reservoir computing" [28, 52] methods, ESNs' theory has been investigated extensively [55, 56], and continues to draw attention to this day, also due to the possibility of implementing reservoirs in non-standard computing environments [51].

As the training of the ESN leverages on linear regression theory, many variants have been proposed in the literature with respect to the standard ℓ_2-regularized cost function of the original paper [22], including training the readout with support vector algorithms [45], imposing additional sparsity constraints via ℓ_1 regularization [15], or composite elastic net penalties [6]. In this paper, we are interested in the more complex problem of training an ESN in a *semi-supervised* learning (SSL) setting, namely, whenever labeled training data is supplemented by additional *unlabeled* data [9]. Although this is a well-known problem in the case of feedforward neural network models, to the best of our knowledge, this has been considered only scarcely in the ESN literature, mostly with respect to specific variations of the original ESN formulation [48], or for unsupervised optimization limited to the reservoir only [28]. However, we argue that an algorithm for exploiting efficiently both labeled and unlabeled data for ESNs would be a very valuable tool, particularly in the case where data is abundant, but costly (or very difficult) to label. As a motivating example, in our experimental results, we will focus on several audio classification tasks, such as event classification for acoustic sensors [3] or automatic music annotation [18]. In the former case, detecting the presence of an event from a stream of audio can be accomplished easily with standard audio processing routines, and accurate beamforming techniques can be applied even with simple array of microphones [42]. On the contrary, correctly labeling each event requires the intervention of a human supervisor that, depending on the situation, could be time-consuming, costly, or downright impossible. Similarly, nowadays the average computer user has a large collection of music files stored in his/her computer, only a subset of which is generally fully tagged/annotated, e.g., by musical genre [39, 50].

In order to derive an efficient algorithm for training any ESN in a SSL context, in this paper, we propose to train the readout using the well-known semi-supervised support vector machine (S^3VM) model [10]. In the S^3VM, we search for an hyperplane having maximum margin, minimizing the error on both labeled and unlabeled data, whose unknown labels are included in the optimization problem as additional variables. Although this results in a mixed integer optimization problem, a wide array of techniques are available for approximating it efficiently [10]. In particular, we leverage on the seminal paper of Chapelle and Zien [12], in order to rewrite the optimization problem as a smooth non-convex problem. In principle, this can be solved by standard gradient descent procedures, which, however, can be suboptimal due to the non-convexity of the training function. As a second contribution of the paper, we customize the algorithm proposed in [41], which is based on recent advances in non-convex optimization theory [14, 17, 44], to solve the problem more efficiently. The algorithm works by solving a series of strongly convex approximations of the original problem, with proven convergence to a stationary point. As a third contribution, always based on the general theory outlined in [17, 44], we show that the proposed algorithm can be straightforwardly decomposed into multiple convex subproblems solvable in parallel, up to one weight per available processor.

Before moving on to the technical content of the paper, we discuss briefly the biological inspiration(s) for the techniques introduced here. In particular, we note that ESNs have attracted a widespread interest in neuroscience, where an equivalent family of techniques, denoted as liquid state machines, are commonly used as a tool for the analysis of neural circuits [29]. In this sense, SSL algorithms, of which the S^3VM is a core member, can be seen as more biologically plausible tools for their training, as we generally learn by a combination of a small number of labeled examples, complemented by a large exposure to unlabeled sensory perceptions. It is not surprising, then, that SSL has found ample recognition in findings of cognitive psychology [58].

The rest of the paper is organized as follows. In the "Echo State Network", we present the standard ESN formulation. The "Semi-supervised Echo State Network" shows how the ESN can be combined with the S^3VM, in order to obtain a semi-supervised criterion for training it. In the subsequent section ("SS-ESN Optimization via Successive Approximations"), we describe an efficient algorithm for solving the optimization problem arising in the semi-supervised ESN, together with a principled way for parallelizing its computation. The "Experimental Evaluation" details an extensive set of experimental results in the context of audio classification, while the "Conclusion" concludes the paper by summarizing the contributions and presenting some final remarks and lines of research.

Notation

In the rest of the paper, vectors are denoted by boldface lowercase letters, e.g., $\mathbf{a}$, while matrices are denoted by boldface uppercase letters, e.g., $\mathbf{A}$. All vectors are assumed to be column vectors unless otherwise specified. The operator $\|\cdot\|_2$

is the standard ℓ_2 norm on an Euclidean space. The spectral radius of a generic matrix $\mathbf{A}$ is $\rho(\mathbf{A}) = \max_i \{|\lambda_i(\mathbf{A})|\}$, where $\lambda_i(\mathbf{A})$ is the ith eigenvector of $\mathbf{A}$. Finally, the notation $a[n]$ is used to denote dependence with respect to a time-instant, both for time-varying signals (in which case n refers to a time-instant) and for elements in an iterative procedure (in which case n is the iteration's index).

Echo State Network

A schematic description of the ESN architecture considered in this paper is shown in Fig. 1. The derivation can be extended trivially to other classes of ESNs, such as multi-layered ones [31]. The input at time n is given by a N_i-dimensional input vector $\mathbf{x}[n]$, which is fed to a N_r-dimensional reservoir whose previous internal state $\mathbf{h}[n-1]$ is updated according to:

$$\mathbf{h}[n] = g(\mathbf{W}_i^r \mathbf{x}[n] + \mathbf{W}_r^r \mathbf{h}[n-1]) \,, \tag{1}$$

where the matrices $\mathbf{W}_i^r \in \mathbb{R}^{N_r \times N_i}$ and $\mathbf{W}_r^r \in \mathbb{R}^{N_r \times N_r}$ are fixed in advance, by drawing their parameters from a predefined probability distribution. $g(\cdot)$ in Eq. 1 is a suitable non-linear function, which we set as $f(\cdot) = \tanh(\cdot)$ in our experiments (although other choices are possible, such as sigmoid functions). In order to obtain a stable behavior, the effect of any previous state should vanish exponentially in time, a property called echo state property (ESP) in the literature [28]. An empirical rule to ensure this is to rescale $\mathbf{W}_r^r$ such that $\rho(\mathbf{W}_r^r) < 1$, albeit more sophisticated procedures are possible [55, 56]. Once the state is updated, the new output of the ESN is computed according to:

$$y[n] = \left(\mathbf{w}_i^o\right)^T \mathbf{x}[n] + \left(\mathbf{w}_r^o\right)^T \mathbf{h}[n] \,, \tag{2}$$

where $\mathbf{w}_i^o \in \mathbb{R}^{N_i}$, $\mathbf{w}_r^o \in \mathbb{R}^{N_r}$ are adapted based on the training data. To increase the overall stability, it is possible to insert a small uniform noise term to the state update in Eq. 1, before computing the non-linear transformation $g(\cdot)$ [22].

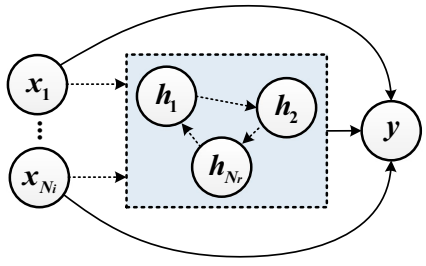

Fig. 1 ESN architecture considered in this paper. Fixed weights are shown with *dashed lines*, while adaptable weights are shown with *solid lines*. The reservoir is shown with a *light blue background*

Owing to the topology of the network, the training procedure for an ESN is also composed of two separate parts. Suppose we are provided with a sequence of inputs $\mathbf{x}[1], \ldots, \mathbf{x}[Q]$, with associated desired responses $d[1], \ldots, d[Q]$. In this paper, we are concerned with binary classification problems, such that each desired output is in the set $\{-1, +1\}$. Multiclass problems can be handled as well, by converting them to multiple binary tasks, e.g., with the use of one-vs-all methods [38]. In the "harvesting" (or "warming") phase, the inputs are provided to the reservoir, and we collect the resulting internal states $\mathbf{h}[1], \ldots, \mathbf{h}[Q]$. For simplicity of notation, we define the (extended) state at time n $\mathbf{s}[n] = \left[\mathbf{x}^T[n]\, \mathbf{h}^T[n]\right]^T$. We stack row-wise the states in the matrix $\mathbf{S}$ and the corresponding responses in the vector $\mathbf{d}$. The readout weights are then computed by solving the following regularized least-squares problem [28]:

$$\mathbf{w}^* = \arg \min_{\mathbf{w} \in \mathbb{R}^{N_i + N_r}} \frac{\lambda}{2} \|\mathbf{S}\mathbf{w} - \mathbf{d}\|_2^2 + \frac{1}{2} \|\mathbf{w}\|_2^2 \,, \tag{3}$$

where $\mathbf{w} = \left[\left(\mathbf{w}_i^o\right)^T \left(\mathbf{w}_r^o\right)^T\right]^T$, and $\lambda \in \mathbb{R}^+$ is a positive scalar weighting the two terms. In Eq. 3, we let λ multiply the error term instead of the regularization factor, as is more common (the two formulations are clearly equivalent up to a rescaling of λ). This is done for having a uniform notation with the next section. A solution of problem (3) can be obtained in closed form as:

$$\mathbf{w}^* = \left(\mathbf{S}^T\mathbf{S} + \frac{1}{\lambda}\mathbf{I}\right)^{-1} \mathbf{S}^T\mathbf{d} \,, \tag{4}$$

where $\mathbf{I}$ is the identity matrix of appropriate dimensionality. Note that, in practice, we may remove the first elements from the sequence, due to the transient state of the reservoir after initialization. Following standard terminology, we refer to them as "wash-out" elements. Additionally, multiple sequences can be handled by stacking the resulting states, after eventually removing the wash-out elements from each of the sequences.

Semi-supervised Echo State Network

In the rest of the paper, we consider the more complex problem of training an ESN in a SSL context, wherein a few of the training sequences do not have associated labels. Although we can easily harvest the corresponding ESN's states for the unlabeled sequences, these cannot be plugged directly in Eq. 3, due to the lack of desired responses. To set the notation for our algorithm, we assume that after harvesting all the sequences (and eventually removing the wash-out

elements), we are left with L labeled states, that we denote as $\mathbf{s}_1, \ldots, \mathbf{s}_L$, L associated desired labels $d_1, \ldots, d_L$, and U unlabeled states, denoted as $\hat{\mathbf{s}}_1, \ldots, \hat{\mathbf{s}}_U$.

Instead of searching for the hyperplane weights $\mathbf{w}^*$ minimizing the least-squares cost in Eq. 3, we propose to search for the optimal hyperplane according to the S^{3}VM algorithm [10], in order to handle the unsupervised states. In particular, the unknown labels $\hat{\mathbf{d}} = \left[\hat{d}_1, \ldots, \hat{d}_U\right]$ are added to the standard SVM optimization problem, resulting in the following cost function:

$$\min_{\mathbf{w},\hat{\mathbf{d}}} \left\{ \frac{\lambda_1}{2} \sum_{i=1}^{L} l\left(d_i, f(\mathbf{s}_i)\right) + \frac{\lambda_2}{2} \sum_{i=1}^{U} l\left(\hat{d}_i, f(\hat{\mathbf{s}}_i)\right) + \frac{1}{2}\|\mathbf{w}\|_2^2 \right\}, \tag{5}$$

where $f(\mathbf{s}) = \mathbf{w}^T\mathbf{s}$, and $l(\cdot,\cdot)$ is the squared hinge loss function:

$$l\left(d, f(\mathbf{s})\right) = \max\left(0, 1 - df(\mathbf{s})\right)^2. \tag{6}$$

We use two different regularization factors λ_1, λ_2 in Eq. 5 in order to weight differently the errors from labeled and unlabeled data. Particularly, for $\lambda_2 = 0$, we recover the standard SVM formulation. Using support vector algorithms for training the ESN is well known in the literature to provide a certain degree of robustness [25, 45], but to the best of our knowledge it was never considered for the specific case of semi-supervised ESNs. It is also customary to include a balancing constraint in the formulation of Eq. 5, in order to obtain a good proportion of positive and negative labels in $\hat{\mathbf{d}}$. We refer to [10, 41] for additional details. We also underline that the general idea of Eq. 5, in which the unknown labels are included in the optimization problem, can be customized also for other algorithms, such as the semi-supervised least-squares SVM [1], or the semi-supervised functional-link network [40]. All of these algorithms can in principle be adapted in this context, and we leave this exploration for future works.

The inclusion of the unknown labels $\hat{\mathbf{d}}$ as variables of the optimization problem makes it a mixed integer optimization problem, whose exact solution can be computed only for relatively small datasets, e.g., using branch and bound algorithms [11]. Numerous researchers have proposed alternative solutions for solving it, including convex relaxations [12, 26], convex-concave procedures [19], and others (see [10] for a comprehensive survey up to 2008).

In this paper, we follow the strategy outlined in [12], stemming from the observation that, for a fixed $\mathbf{w}$, the optimal $\hat{\mathbf{d}}$ is given element-wise in closed form by $\hat{d}_i = \text{sign}(\mathbf{w}^T\hat{\mathbf{s}}_i)$, $i = 1, \ldots, U$. Exploiting this, it is possible to devise a *smooth* approximation of the cost function in (5). Particularly, we replace the hinge loss over the unknown labels with the squared exponential approximation given by $\exp\left\{-sf(\mathbf{s})^2\right\}$, $s > 0$. In the following, we choose in particular $s = 5$, as suggested by [10]. A visual example of the approximation is given in Fig. 2 for this particular choice of parameterization. Summarizing, the resulting ∇S^3VM optimization problem writes as:

$$\min_{\mathbf{w}} \left\{ \frac{\lambda_1}{2} \sum_{i=1}^{L} l\left(d_i, f(\mathbf{s}_i)\right) + \frac{\lambda_2}{2} \sum_{i=1}^{U} \exp\left\{-5f(\hat{\mathbf{x}}_i)^2\right\} + \frac{1}{2}\|\mathbf{w}\|_2^2 \right\}. \tag{7}$$

Problem (7) is now a smooth problem (as the integer variables have been removed), and non-convex due to the presence of the squared exponential term. It can already be solved using standard first-order procedures, as in the original paper [12]. In the following section, however, we propose a novel algorithm which is able to efficiently exploit the structure of problem (7) and can eventually be parallelized by exploiting the presence of multiple processors.

SS-ESN Optimization via Successive Approximations

Formulation of the Algorithm

One intriguing property of the cost function (7) is that it is composed by a sum of three terms, two of which

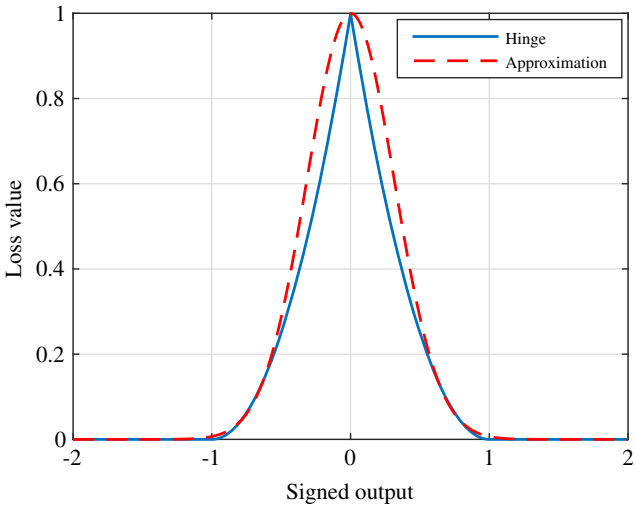

Fig. 2 After fixing the vector $\mathbf{w}$, by simple algebra we obtain $\max\left(0, 1 - \hat{d}_i f(\hat{\mathbf{s}}_i)\right)^2 = \max\left(0, 1 - |f(\hat{\mathbf{s}}_i)|\right)^2$. This is shown in *blue* for varying values of $f(\hat{\mathbf{s}}_i)$, while in *dashed red* we show the approximation given by $\exp\left\{-5f(\hat{\mathbf{s}}_i)^2\right\}$

are convex, while the second one (the squared exponential) is non-convex. A recent line of research in non-convex optimization is dedicated to exploiting this kind of "structural" information of the cost function in order to devise algorithms with fast (provable) convergence to a stationary point, resulting in what we denote as the successive convex approximation (SCA) framework [17]. In this section, we only provide the basic information required to derive our algorithm, and we refer the interested reader to [14, 17, 44] for more details on the general theory.

The basic algorithm works by solving a series of successive strongly convex approximations of the original problem. For simplicity, we define the following shorthands:

$$l(\mathbf{w}) = \frac{\lambda_1}{2} \sum_{i=1}^{L} l\left(d_i, f(\mathbf{s}_i)\right) , \tag{8}$$

$$g(\mathbf{w}) = \frac{\lambda_2}{2} \sum_{i=1}^{U} \exp\left\{-5 f(\hat{\mathbf{s}}_i)^2\right\} , \tag{9}$$

$$r(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_2^2 . \tag{10}$$

Also, define $U(\mathbf{w}) = l(\mathbf{w}) + g(\mathbf{w})$, and denote by $\mathbf{w}[k]$ the estimate of the optimal weights at iteration k. We are looking for a surrogate function $\tilde{U}(\mathbf{w}; \mathbf{w}[k])$ of $U(\mathbf{w})$ with the following properties:

(P1) $\tilde{U}(\cdot; \mathbf{w}[k])$ is convex and continuously differentiable for any choice of $\mathbf{w}[k]$.

(P2) $\nabla_{\mathbf{w}} \tilde{U}(\mathbf{w}; \mathbf{w}[k]) = \nabla_{\mathbf{w}} U(\mathbf{w})$ for any possible choice of $\mathbf{w}[k]$.

(P3) $\nabla_{\mathbf{w}} \tilde{U}(\mathbf{w}; \cdot)$ is Lipschitz continuous for any choice of $\mathbf{w}$.

The three properties ensure that the approximation is relatively simple and preserves the first-order properties of the original cost function. Similarly to [41], one straightforward way to build the surrogate function is to consider the first-order linearization of $g(\mathbf{w})$ around $\mathbf{w}[k]$:

$$\tilde{g}(\mathbf{w}; \mathbf{w}[k]) = g(\mathbf{w}[k]) + \nabla_{\mathbf{w}[k]} g^T(\mathbf{w}[k]) \left(\mathbf{w} - \mathbf{w}[k]\right) . \tag{11}$$

Summing back the regularization term, we obtain the following strongly convex optimization problem at iteration k:

$$\mathbf{z} = \arg\min_{\mathbf{w}} \left\{ l(\mathbf{w}) + \tilde{g}(\mathbf{w}; \mathbf{w}[k]) + \frac{1}{2} \mathbf{w}^T \mathbf{w} \right\} . \tag{12}$$

After solving the problem, the current estimate is updated according to the following convex combination:

$$\mathbf{w}[k+1] = \mathbf{w}[k] + \eta[k] \left(\mathbf{z} - \mathbf{w}[k]\right) , \tag{13}$$

where $\eta[k]$ is an iteration-dependent step-size. We have the following convergence theorem.

Theorem 1 *Let* $\{\mathbf{w}[k]\}_n$ *be the sequence generated by iteratively applying* Eqs. 12 and 13. *Suppose that the step-size sequence* $\{\eta[k]\}_n$ *is chosen so that* $\eta[k] \in (0, 1]$, *for all k, and:*

$$\sum_{k=0}^{\infty} \eta[k] = \infty \text{ and } \sum_{k=0}^{\infty} \eta^2[k] < \infty. \tag{14}$$

Then, the sequence $\{\mathbf{w}[k]\}_n$ *converges to a stationary solution of Problem* (7) *in a finite number of steps.*

Proof See [17]. □

The proposed algorithm has a sort of quadratic complexity with respect to a standard gradient descent, since we need to solve a series of surrogate (sub)problems. However, strong convexity of the surrogate problems means that we can exploit a wide range of very efficient procedures for their solution [34]. Moreover, at every iteration we can consider a "warm start" of the optimization algorithm from the previous estimate $\mathbf{w}[k]$ which, especially on the final stages of optimization, can provide an enormous boost in convergence time. Third, the original theory in [17] also admits *inexact* solutions of the surrogate problems, as long as the approximation error satisfies some boundedness constraints (see [17] for more details). Last but not least, as we show in the following subsection, the algorithm can be easily parallelized up to one weight per available processor.

Remark 1 There is a sense in which linearizing only the non-convex terms in Eq. 7 is the most basic way of obtaining a strongly convex approximation, since it allows us to keep as much as possible of its original structure, without requiring the expensive computation of second-order information on its derivatives. However, it is possible to make different design choices, resulting in additional customizations of the general framework presented here, as long as properties (P1)–(P3) are satisfied. As an example, we may consider to substitute also $l(\mathbf{w})$ with its first-order approximation $\tilde{l}(\mathbf{w})$, similar to Eq. 11. In this case, it is straightforward to show that the solution of the surrogate problem is obtained in closed form as:

$$\mathbf{z}' = -\left(\nabla_{\mathbf{w}[k]} \tilde{g}(\mathbf{w}[k]) + \nabla_{\mathbf{w}[k]} \tilde{l}(\mathbf{w}[k])\right) . \tag{15}$$

This simply means that, by only keeping first-order information on the original problem, the best we can do is to move towards the steepest descent direction, as prescribed by Eq. 15. Combining this with Eq. 13, the resulting optimization scheme is similar to a standard steepest descent procedure, but it moves by performing successive convex combinations with respect to the previous estimate:

$$\mathbf{w}[k+1] = (1 - \mu[k])\, \mathbf{w}[k] + \mu[k] \mathbf{z}' . \tag{16}$$

Clearly, the convergence properties of Theorem 1 apply also in this case.

Parallel Computation of the Surrogate Function

Algorithm 1 Parallel semi-supervised ESN (SS-ESN) using the SCA framework.

Input: Regularization factors λ_1, λ_2, maximum number of iterations T.

Initialization:
  Initialize the reservoir (see Section 2).
  Initialize output weights $\mathbf{w}[1] = \mathbf{0}$.
Harvesting:
  Harvest states $\mathbf{s}_1, \ldots, \mathbf{s}_L$ for labeled sequences (removing wash-out elements).
  Harvest states $\hat{\mathbf{s}}_1, \ldots, \hat{\mathbf{s}}_U$ for unlabeled sequences (removing wash-out elements).
Main loop:
  for n from 1 to T **do**
    for p from 1 to P **do in parallel**
      Solve the local optimization problem in Eq. (19).
    end for
    Compute $\mathbf{w}[k+1]$ using (13).
  end for

One interesting aspect of the previously detailed framework is that it leads itself automatically to a parallel decomposition of the optimization part. Particularly, assume we have available P agents for executing the computation, e.g., P different processors on the same machine, or P different machines over a cloud. Then, define a block-partitioning $\mathbf{w}_1, \ldots, \mathbf{w}_P$ of $\mathbf{w}$ such that $\bigcup_{p=1}^{P} \mathbf{w}_p = \mathbf{w}$, and denote as $\mathbf{w}_{-p}$ the union of all blocks except the pth one. We define a new surrogate cost as the sum of P local costs as follows:

$$\hat{U}(\mathbf{w}; \mathbf{w}[k]) = \sum_{p=1}^{P} \hat{U}(\mathbf{w}_p; \mathbf{w}_{-p}[k]) \,, \tag{17}$$

where each $\hat{U}(\mathbf{w}_p; \mathbf{w}_{-p}[k])$ is a surrogate function satisfying (P1)–(P3) in the variable $\mathbf{w}_p$. Then, the original optimization problem can be solved by optimizing P *independent* subproblems as follows:

$$\begin{aligned} &\arg\min_{\mathbf{z}} \left\{ \hat{U}(\mathbf{w}; \mathbf{w}[k]) + \frac{1}{2}\mathbf{w}^T\mathbf{w} \right\} \\ &\quad = \bigcup_{p=1}^{P} \arg\min_{\mathbf{z}_p} \left\{ \hat{U}(\mathbf{w}_p; \mathbf{w}_{-p}[k]) + \frac{1}{2}\mathbf{w}_p^T\mathbf{w}_p \right\} . \end{aligned} \tag{18}$$

If we use the same kind of surrogate function as in the last subsection, by defining the pth residual as $r_p = \mathbf{w}_{-p}^T \mathbf{s}_{-p}$, the pth surrogate cost writes as (omitting constant terms):

$$\begin{aligned} \arg\min_{\mathbf{z}} \; = \; & \frac{\lambda_1}{2} \sum_{i=1}^{L} \max\left(0, 1 - d_i \left(\mathbf{w}_p^T \mathbf{x}_p + r_p\right)\right) \\ & + \nabla_{\mathbf{w}_p} g^T(\mathbf{w}[k]) \mathbf{w}_p + \frac{1}{2}\mathbf{w}_p^T \mathbf{w}_p \,. \end{aligned} \tag{19}$$

It is straightforward to see from Eq. 19 that we can obtain a potential linear speedup in the optimization process, both in terms of matrix-vector multiplications and vector-vector multiplications. The algorithm obeys the same convergence properties as those outlined in Theorem 1. For completeness, the overall algorithm is summarized in Algorithm 1. Note that for $P = 1$, with $\mathbf{w}_1 = \mathbf{w}$, we recover the formulation exposed in the previous subsection. The memory requirement of the algorithm, as most batch algorithms for ESNs, is given by the need of storing the $(L+U) \times (N_i + N_r)$ state matrix of the reservoir, along with the corresponding vector of labels. The time requirement, instead, is dominated at every iteration by the need of solving P subproblems as in Eq. 19. Each of them is a strongly convex quadratic problem of dimensionality, roughly $\lfloor \frac{N_i+N_r}{P} \rfloor$, whose complexity can be customized to the available computational power by suitably varying P.

Experimental Evaluation

Experimental Setup

In this section, we evaluate the proposed algorithm on a set of different audio classification problems. In all cases, the original audio files are preprocessed by subdividing them in frames of 0.1 s (0.2 s in the second case), with a 50 % overlap among them. Then, a comprehensive set of 30 audio features is extracted from each frame using the MIRToolbox in MATLAB [24]. These include the root mean square energy, spectral centroid and variance, the spectral flux, the zero-crossing count, 12 chroma features (corresponding to the distribution of energy along 12 pitch classes), and 13 Mel-frequency cepstral coefficients (MFCCs). For a more complete description of the features, we refer to the documentation of the toolbox[1] and to [18]. After extraction, all features are normalized with an affine transformation to lie in the [0, 1] range.

For every run, training data sequences are split according to a fivefold cross-validation procedure for testing purposes. The fivefold cross-validation procedure is repeated 25 times

[1] https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox [last visited on November 19, 2016]

by varying the ESN initialization and the data partitioning, and the errors for every iteration and every fold are collected. To compute the error, we run each trained ESN on the test sequences and gather the predicted outputs $\tilde{y}_1, \ldots, \tilde{y}_K$, where K is the number of testing samples after removing the wash-out elements from the test sequences. Then, we compute the misclassification error, defined as:

$$\text{ME} = \sum_{i=1}^{K} \mathbb{I}\left\{\text{sign}\left(\tilde{y}_i\right) - \text{sign}\left(d_i\right)\right\}, \tag{20}$$

where $\mathbb{I}$ is the indicator function and sign$(\cdot)$ takes the sign of its argument.

All algorithms share the same ESN architecture, which we detail briefly here. First, we choose a default reservoir's size of $N_r = 300$, which was found to work well in all situations. Then, we make the following design choices:

- The matrix $\mathbf{W}_i^r$, connecting the input to the reservoir, is initialized as a full matrix, with entries assigned from the uniform distribution $[-0.1, 0.1]$.
- The internal reservoir matrix $\mathbf{W}_r^r$ is initialized from the uniform distribution $[-1, +1]$. Then, on average 75 % of its connections are set to 0, to encourage sparseness. To preserve stability, the matrix is rescaled so as to have a desired spectral radius ρ^*, whose value is chosen in order to obtain the highest performance depending on the task.

We also insert uniform noise in the state update (1) of the reservoir, sampled uniformly in the interval $\left[0, 10^{-3}\right]$. For all datasets, we supplement the original input with an additional constant unitary input, as is standard practice in ESNs' implementations [28]. MATLAB code to repeat the experiments is available on the web under BSD license.[2] Simulations were performed on MATLAB R2015a, on a 64bit operative system, using an Intel® Core™ i5-3330 CPU with 3 GHZ and 16 GB of RAM.

Experiment 1: Indoor Acoustic Event Detection

As a first use case, we consider the problem of event recognition for an acoustic sensor in an indoor environment. It is known that this is a fundamental problem for, e.g., context-aware applications to be deployed on mobile devices [2, 16]. For the simplified purpose of this paper, we suppose that the sound has been correctly separated from its environment, e.g., by the use of beamforming techniques [42], and it only needs to be classified framewise according to the features detailed in the previous section. Particularly, we consider the two classes "knocking door" and "washing machine" taken from the dataset in [5], which can also be downloaded from the web.[3] There are 40 recorded sounds from each class, each of roughly 3 s. After preprocessing, we obtain approximately 60 audio frames for each sequence, for which we discard the first five as wash-out.

We compare a standard ESN trained with Eq. 3, with a semi-supervised ESN trained using the algorithm described in the "SS-ESN Optimization via Successive Approximations". The desired spectral radius for this experiment is set to $\rho^* = 0.5$. For simulating a situation of low availability of (labeled) training data, we randomly keep only a small portion of the samples in the training set. In particular, we experiment with $L = \{10, 25, 50, 100, 250\}$, while the amount of unlabeled data is kept fixed at $U = 1000$. For the proposed algorithm, we consider the following rule for choosing the step-size:

$$\mu[k+1] = \frac{\mu[0]}{k^{\delta}}, \tag{21}$$

where $\mu[0]$ and δ are set empirically to provide a smooth convergence behavior. In particular, in this experiment, we set $\mu[0] = 0.2$ and $\delta = 0.1$. For this first experiment, we do not consider parallelization, so we set $P = 1$. The internal surrogate problem in Eq. 12 is solved using a standard steepest gradient descent procedure, where the step-size is chosen according to Eq. 21, with $\mu[0] = 0.5$, and $\delta = 0.55$. We set a maximum of 250 iterations for the algorithm and of 50 iterations for the internal solver. The regularization factors are set by performing an internal threefold cross-validation using all the available training data, in the exponential interval 10^i, $i = -10, \ldots, 10$.

The results in terms of testing error of the resulting ESNs are shown in Fig. 3, where the results for the centralized ESN (ESN in the figure) are shown with a blue line, while the results for the semi-supervised ESN (SS-ESN in the figure) are shown with a red line. We can see clearly that, although the standard ESN suffers from the low availability of labeled data, the SS-ESN is able to consistently improve the performance to the optimal one, which is around 0.5 % of frames misclassified at every fold, showing the usefulness of the proposed method.

Next, we analyze the convergence speed of the algorithm. We focus on the case $L = 10$, although the results are similar for other configurations. To this end, we show the evolution of the ℓ_2 norm of the gradient of Eq. 7 per iteration, comparing the SCA algorithm detailed in the "SS-ESN Optimization via Successive Approximations" with a standard steepest descent procedure, whose step-size is chosen according to rule (21). Parameters are set in order to provide the fastest (smooth) convergence, as $\mu[0] = 0.1$, $\delta = 0.1$. Results of this comparison are shown in Fig. 4, where it can

[2] https://bitbucket.org/ispamm/semi-supervised-esn [last visited on November 19, 2016]

[3] http://sound.natix.org/ [last visited November 19, 2016]

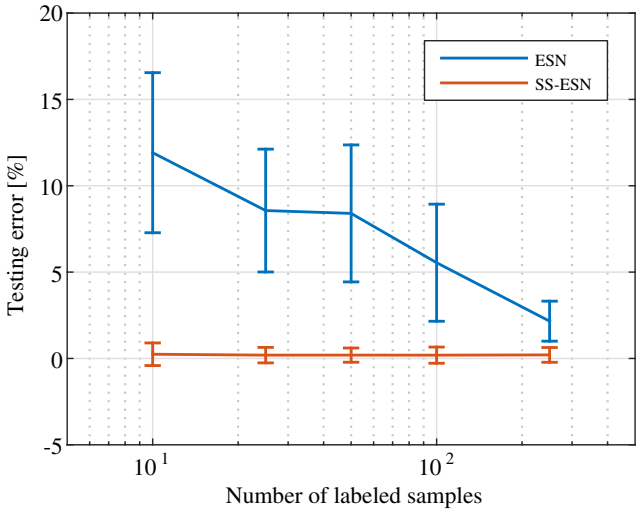

Fig. 3 Test error (mean and 1 standard deviation) for the standard ESN and semi-supervised ESN, when varying the amount of labeled data in $L = \{10, 25, 50, 100, 250\}$. Unlabeled data is kept fixed at $U = 1000$

be seen that the SCA procedure is able to converge towards a stationary solution faster (in terms of iterations) than its competitor, with a gradient norm of 10^{-5} in approximately 100 iterations. Clearly, this comes at the cost of a slightly higher computational need. In particular, in this example, the gathering phase required ≈ 0.9 s (which is common to all algorithms), while the proposed algorithm required 2.7 s against 1 s of the standard gradient descent. However, the proposed framework can be easily customized in a variety of ways in order to improve convergence, which we explore here and in the following section.

The parallelization strategy of the "Parallel Computation of the Surrogate Function", which is an obvious choice, will

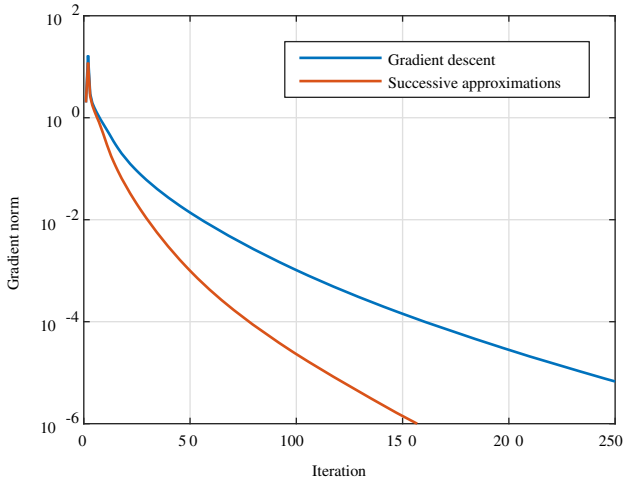

Fig. 4 Evolution of the gradient norm when training the SS-ESN, both with standard gradient descent, and with the newly proposed method based on the SCA framework

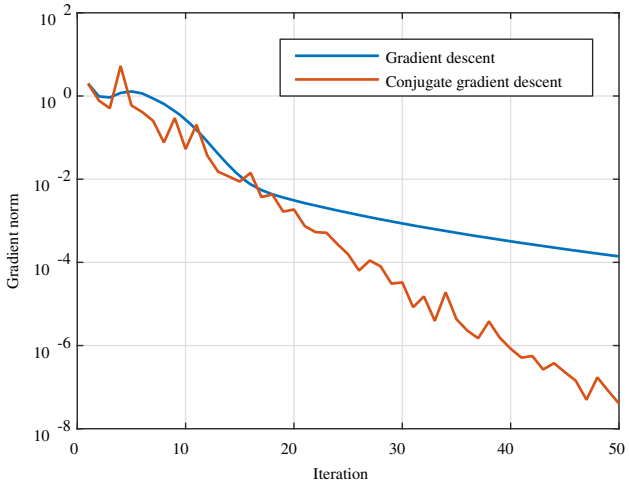

Fig. 5 Evolution of the gradient norm when solving the surrogate problem in Eq. 12 for $k = 1$, using both a standard gradient descent solver, and a conjugate gradient optimization algorithm

be explored in the next experiment, where we consider a larger training set. Here, we focus on the possibility of solving the surrogate problem in Eq. 12 with a more elaborate procedure than the classical steepest descent. To this end, we compare with a freely available MATLAB implementation of the Polack-Ribiere variant of the nonlinear conjugate gradient optimization algorithm by C.E. Rasmussen [37].[4] In Fig. 5, we show the evolution of the ℓ_2 norm of the gradient of Eq. 12 in the first iteration of the algorithm, i.e., $k = 1$. We see that, although progress in the beginning is similar, the conjugate gradient algorithm is able to keep an exponential convergence behavior, with a gradient norm of 10^{-7} in less than 50 iterations. By fixing an internal stopping criterion whenever the gradient's norm is under 10^{-5} for both algorithms, running the proposed algorithm with the conjugate gradient procedure obtains a final training time of 0.8 s, i.e., the resulting process is more than $3\times$ faster.

Experiment 2: Acoustic Scene Classification

As a second test case for the proposed algorithm, we consider a problem of acoustic scene classification [3, 57], whereas the task is for an embedded device to recognize its surrounding environment starting from an acoustic sensory information. To this end, we consider a binary classification task extracted from the 2014 IEEE AASP challenge [46], where we wish to discriminate among two different busy environments, namely "busy street" and "open air market". We have 10 different recordings (of 30 s each) from each environment, which we segment in frames of 0.2 s

[4] http://learning.eng.cam.ac.uk/carl/code/minimize/ [last accessed November 19, 2016]

Table 1 Final test error and training time for the standard ESN and the SS-ESN, for two different amounts of available labeled data

Data		ESN	SS-ESN
$L = 10$	Test error:	0.25 ± 0.07	0.19 ± 0.07
	Tr. time:	2.46 ± 0.00	10.27 ± 0.92
$L = 25$	Test error:	0.22 ± 0.07	0.16 ± 0.07
	Tr. time:	2.37 ± 0.00	10.59 ± 0.57

with 50 % overlap, and we extract the same features as the previous section. We obtain 595 training sample from each sequence, of which we discard 80 as wash-out elements. We follow the same procedure as before for comparing the two algorithms, and based on the aforementioned analysis, we select the conjugate optimization algorithm to solve each inner surrogate problem. For the ESN, we found a slightly higher spectral radius of $\rho^* = 0.7$ to provide the highest performance.

The results for two different amounts of labeled data, i.e., $L = 10$ and $L = 25$, are given in Table 1, where the first column shows the results for the standard ESN, and the second column shows the results for the proposed SS-ESN. As before, the use of the semi-supervised routine strongly outperforms the standard ESN, with a final test error which is approximately 6 % lower than the competitor in both settings. Clearly, this comes at the cost of a higher training time, since the ridge regression routine only requires the inversion of a relatively small matrix, while the semi-supervised routine exploits the additional 1000 unlabeled samples provided to it.

One possibility to improve this aspect is to employ the parallelization strategy outlined before. To this end, we consider the use of multiple processors to which the surrogate

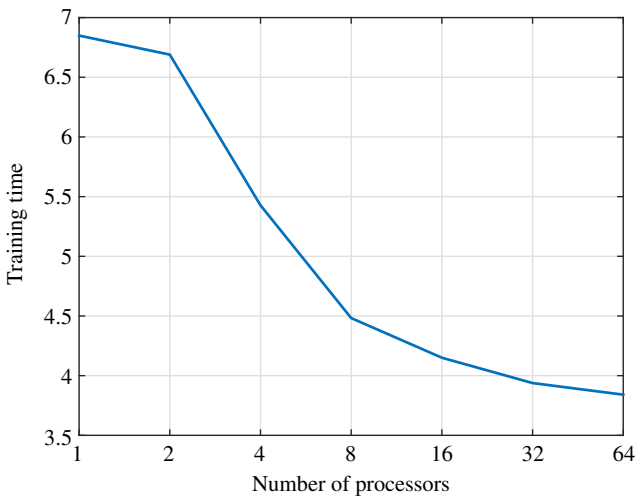

Fig. 6 Training time (not considering gathering time of the input and output sequences) spent on solving the optimization problem for different number of (virtual) processors

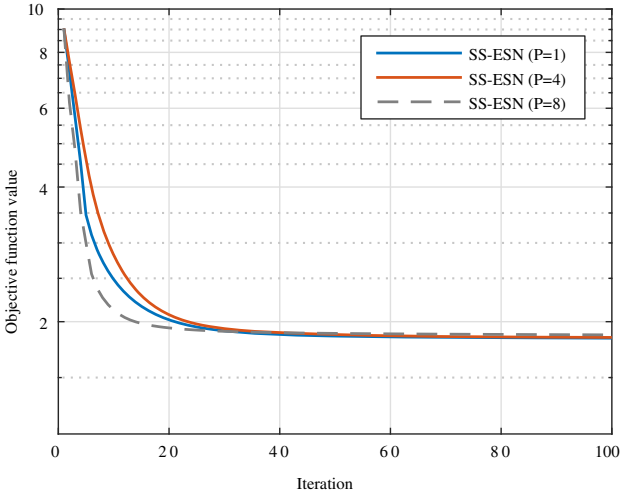

Fig. 7 Evolution of the objective function for three different values of P

(sub)problems are allocated. In particular, we experiment with an exponential range of processors 2^i, with $i = \{0, \ldots, 6\}$. For simplicity, we keep the same configuration (in terms of step-size generation) of the non-parallel version. The resulting training time (without considering the constant time of gathering the input and output sequences, and removing the wash-out elements), is provided in Fig. 6. We see that, even with only four processors, the resulting algorithm is already 22 % faster, while using eight processors it is almost twice as fast. After a certain number of available processors, the relative decrease in training time is slower, as the computational cost become superseded by the overhead of solving too many (smaller) optimization problems.

To further visualize the convergence behavior of the parallel algorithm, we plot in Fig. 7 the evolution of the global objective function (7) for three different choices of the number of processors (i.e., $P = 1$ in blue, $P = 4$ in red, and $P = 8$ in dashed gray). We see that the change brought by solving P problems instead of a single one does not depend linearly on the number of processors. In particular, with the specific configuration used in this experiment, having four processors brings a small reduction in convergence time (as shown by the red line), while having eight processors (meaning eight parallel optimization problems to be solved) is actually able to speed the overall convergence, as shown by the dashed gray line.

Conclusion

In this paper, we have introduced a principled algorithm for training an ESN in a semi-supervised setting, where only a subset of the available training sequences are labeled. This is obtained by combining the standard ESN with a

relaxation of the semi-supervised SVM, giving rise to a non-convex optimization criterion. As a second contribution, we proposed an innovative algorithm to solve the problem, by exploiting a recently proposed framework based on solving successive convex approximations of the original problem. We have shown how the algorithm can be easily decomposed in multiple subproblems solvable in parallel and, finally, we have applied the resulting algorithm to two different audio classification tasks, leading the way to further work on the application of the proposed techniques to real-world sensor applications in intelligent environments [8], sentiment analysis [35], and several others.

Despite our focus has been on audio tasks, we believe the resulting algorithm has the potential of being highly useful in a variety of realistic applications, including many where ESN and related reservoir computing techniques have been shown to perform optimally (e.g., short-term forecasting in semi-supervised scenarios). Additionally, we are interested in designing alternative strategies for the semi-supervised ESN, possibly hinging on the large amount of work to be found in the literature. As an example, a potential line of research could be to develop algorithms based on the Laplacian SVM [4], in which it is assumed that the input data lies on a lower dimensional manifold, such that close points in the manifold have similar outputs [30]. In the context of ESNs, one possibility would be to consider different embeddings for the input space and the reservoir's states, in order to impose multiple regularization terms depending on the application.

Compliance with Ethical Standards

Conflict of Interests The authors declare that they have no conflict of interest.

Ethical Approval This article does not contain any studies with human or animal subjects performed by any of the authors.

References

1. Adankon MM, Cheriet M, Biem A. Semisupervised least squares support vector machine. IEEE Trans Neural Netw. 2009;20(12):1858–1870.
2. Bacciu D, Barsocchi P, Chessa S, Gallicchio C, Micheli A. An experimental characterization of reservoir computing in ambient assisted living applications. Neural Comput & Applic. 2014;24(6):1451–1464.
3. Barchiesi D, Giannoulis D, Stowell D, Plumbley MD. Acoustic scene classification: Classifying environments from the sounds they produce. IEEE Signal Process Mag. 2015;32(3):16–34.
4. Belkin M, Niyogi P, Sindhwani V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. J Mach Learn Res. 2006;7:2399–2434.
5. Beltrán J, Chávez E, Favela J. Scalable identification of mixed environmental sounds, recorded from heterogeneous sources. Pattern Recogn Lett. 2015;68:153–160.
6. Bianchi FM, Scardapane S, Uncini A, Rizzi A, Sadeghian A. Prediction of telephone calls load using echo state network with exogenous variables. Neural Netw. 2015;71:204–213.
7. Campolucci P, Uncini A, Piazza F, Rao BD. On-line learning algorithms for locally recurrent neural networks. IEEE Trans Neural Netw. 1999;10(2):253–271.
8. Castillo JC, Castro-González Á, Fernández-Caballero A, Latorre JM, Pastor JM, Fernández-Sotos A, Salichs MA. Software architecture for smart emotion recognition and regulation of the ageing adult. Cogn Comput. 2016;8(2):357–367.
9. Chapelle O, Schölkopf B, Zien A. Semi-supervised learning MIT Press Cambridge. 2006.
10. Chapelle O, Sindhwani V, Keerthi S. Optimization techniques for semi-supervised support vector machines. J Mach Learn Res. 2008;9:203–233.
11. Chapelle O, Sindhwani V, Keerthi SS. Branch and bound for semi-supervised support vector machines. Advances in neural information processing systems; 2006. p. 217–224.
12. Chapelle O, Zien A. Semi-supervised classification by low density separation. Proceedings of the tenth international workshop on artificial intelligence and statistics; 2005. p. 57–64.
13. Chatzis SP, Demiris Y. Echo state gaussian process. IEEE Trans Neural Netw. 2011;22(9):1435–1445.
14. Di Lorenzo P, Scutari G. NEXT: In-network nonconvex optimization. IEEE Transactions on Signal and Information Processing over Networks. 2016;2(2):120–136.
15. Dutoit X, Schrauwen B, Van Campenhout J, Stroobandt D, Van Brussel H, Nuttin M. Pruning and regularization in reservoir computing. Neurocomputing. 2009;72(7):1534–1546.
16. Eronen AJ, Peltonen VT, Tuomi JT, Klapuri AP, Fagerlund S, Sorsa T, Lorho G, Huopaniemi J. Audio-based context recognition. IEEE Trans Audio Speech Lang Process. 2006;14(1):321–329.
17. Facchinei F, Scutari G, Sagratella S. Parallel selective algorithms for nonconvex big data optimization. IEEE Trans Signal Process. 2015;63(7):1874–1889.
18. Fu Z, Lu G, Ting KM, Zhang D. A survey of audio-based music classification and annotation. IEEE Trans Multimedia. 2011;13(2):303–319.
19. Fung G, Mangasarian OL. Semi-supervised support vector machines for unlabeled data classification. Optimization methods and software. 2001;15(1):29–44.
20. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1–32.
21. Jaeger H. The echo state approach to analysing and training recurrent neural networks. Tech. rep., GMD Report 148 German National Research Center for Information Technology. 2001.
22. Jaeger H. Adaptive nonlinear system identification with echo state networks. Advances in Neural Information Processing Systems; 2002. p. 593–600.
23. Jaeger H., Haas H. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science. 2004;304(5667):78–80.
24. Lartillot O, Toiviainen P. A matlab toolbox for musical feature extraction from audio. International Conference on Digital Audio Effects, pp. 237–244; 2007.
25. Li D, Han M, Wang J. Chaotic time series prediction based on a novel robust echo state network. IEEE Transactions on Neural Networks and Learning Systems. 2012;23(5):787–799.
26. Li YF, Tsang IW, Kwok JT. Convex and scalable weakly labeled SVMs. J Mach Learn Res. 2013;14:2151–2188.

27. Lin X, Yang Z, Song Y. Short-term stock price prediction based on echo state networks. Expert Systems with Applications. 2009;36(3):7313–7317.
28. Lukoševičius M, Jaeger H. Reservoir computing approaches to recurrent neural network training. Computer Science Review. 2009;3(3):127–149.
29. Maass W, Natschläger T, Markram H. Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. 2002;14(11):2531–2560.
30. Malik ZK, Hussain A, Wu J. An online generalized eigenvalue version of Laplacian eigenmaps for visual big data. Neurocomputing. 2016;173:127–136.
31. Malik ZK, Hussain A, Wu QJ. Multilayered echo state machine: a novel architecture and algorithm. IEEE Transactions on Cybernetics. 2016:1–14. In press.
32. Martens J, Sutskever I. Learning recurrent neural networks with Hessian-free optimization. Proceedings of the 28th International Conference on Machine Learning (ICML'11); 2011. p. 1033–1040.
33. Meftah B, Lézoray O, Benyettou A. Novel approach using echo state networks for microscopic cellular image segmentation. Cogn Comput. 2016;8(2):237–245.
34. Nesterov Y. Introductory lectures on convex optimization: a basic course Springer Science & Business Media. 2013.
35. Pandarachalil R, Sendhilkumar S, Mahalakshmi G. Twitter sentiment analysis for large-scale data: an unsupervised approach. Cogn Comput. 2015;7(2):254–262.
36. Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks. Proceedings of the 30th International Conference on Machine Learning (ICML'12) (2); 2012. p. 1310–1318.
37. Rasmussen CE. Gaussian processes for machine learning MIT Press. 2006.
38. Rifkin R, Klautau A. In defense of one-vs-all classification. J Mach Learn Res. 2004;5:101–141.
39. Scardapane S, Comminiello D, Scarpiniti M, Uncini A. Music classification using extreme learning machines. Proceedings of the 2013 IEEE International Symposium on Image and Signal Processing and Analysis (ISPA'13), pp. 377–381; 2013.
40. Scardapane S, Comminiello D, Scarpiniti M, Uncini A. A semi-supervised random vector functional-link network based on the transductive framework. Inf Sci. 2015;364–365:156—-166.
41. Scardapane S, Fierimonte R, Di Lorenzo P, Panella M, Uncini A. Distributed semi-supervised support vector machines. Neural Netw. 2016;80:43–52.
42. Scardapane S, Scarpiniti M, Bucciarelli M, Colone F, Mansueto MV, Parisi R. Microphone array based classification for security monitoring in unstructured environments. AEU-Int J Electron C. 2015;69(11):1715–1723.
43. Scardapane S, Wang D, Panella M. A decentralized training algorithm for echo state networks in distributed big data applications. Neural Netw. 2016;78:65—74.
44. Scutari G, Facchinei F, Song P, Palomar DP, Pang JS. Decomposition by partial linearization: Parallel optimization of multi-agent systems. IEEE Transactions on Signal Processing. 2014;62(3):641–656.
45. Shi Z, Han M. Support vector echo-state machine for chaotic time-series prediction. IEEE Trans Neural Netw. 2007;18(2):359–372.
46. Stowell D, Giannoulis D, Benetos E, Lagrange M, Plumbey M. Detection and classification of audio scenes and events. IEEE Trans Multimedia. 2015;17(10):1733–1746.
47. Tong M. H, Bickett AD, Christiansen EM, Cottrell GW. Learning grammatical structure with echo state networks. Neural Netw. 2007;20(3):424–432.
48. Trentin E, Scherer S, Schwenker F. Emotion recognition from speech signals via a probabilistic echo-state network. Pattern Recogn Lett. 2015;66:4–12.
49. Triefenbach F, Jalalvand A, Demuynck K, Martens JP. Acoustic modeling with hierarchical reservoirs. IEEE Trans Audio Speech Lang Process. 2013;21(11):2439–2450.
50. Tzanetakis G, Cook P. Musical genre classification of audio signals. IEEE Transactions on Speech and Audio Processing. 2002;10(5):293–302.
51. Vandoorne K, Mechet P, Van Vaerenbergh T, Fiers M, Morthier G, Verstraeten D, Schrauwen B, Dambre J, Bienstman P. Experimental demonstration of reservoir computing on a silicon photonics chip. Nat Commun. 2014;5:1–6.
52. Verstraeten D, Schrauwen B, d'Haene M, Stroobandt D. An experimental unification of reservoir computing methods. Neural Netw. 2007;20(3):391–403.
53. Wang P, Song Q, Han H, Cheng J. Sequentially supervised long short-term memory for gesture recognition. Cogn Comput. 2016:1–10. In press.
54. Werbos PJ. Backpropagation through time: what it does and how to do it. Proc IEEE. 1990;78(10):1550–1560.
55. Yildiz IB, Jaeger H, Kiebel SJ. Re-visiting the echo state property. Neural Netw. 2012;35:1–9.
56. Zhang B, Miller DJ, Wang Y. Nonlinear system modeling with random matrices: echo state networks revisited. IEEE Transactions on Neural Networks and Learning Systems. 2012;23(1):175–182.
57. Zhao J, Du C, Sun H, Liu X, Sun J. Biologically motivated model for outdoor scene classification. Cogn Comput. 2015;7(1):20–33.
58. Zhu X, Goldberg AB. Introduction to semi-supervised learning. Synthesis lectures on artificial intelligence and machine learning. 2009;3(1):1–130.