Echo State Gaussian Process

Article *in* IEEE Transactions on Neural Networks · July 2011

DOI: 10.1109/TNN.2011.2162109 · Source: PubMed

2 authors:

Sotirios P Chatzis
Cyprus University of Technology
76 PUBLICATIONS **603** CITATIONS

SEE PROFILE

Yiannis Demiris
Imperial College London
188 PUBLICATIONS **2,380** CITATIONS

SEE PROFILE

Echo State Gaussian Process

Sotirios P. Chatzis, *Member, IEEE*, and Yiannis Demiris, *Senior Member, IEEE*

***Abstract*— Echo state networks (ESNs) constitute a novel approach to recurrent neural network (RNN) training, with an RNN (the reservoir) being generated randomly, and only a readout being trained using a simple computationally efficient algorithm. ESNs have greatly facilitated the practical application of RNNs, outperforming classical approaches on a number of benchmark tasks. In this paper, we introduce a novel Bayesian approach toward ESNs, the echo state Gaussian process (ESGP). The ESGP combines the merits of ESNs and Gaussian processes to provide a more robust alternative to conventional reservoir computing networks while also offering a measure of confidence on the generated predictions (in the form of a predictive distribution). We exhibit the merits of our approach in a number of applications, considering both benchmark datasets and real-world applications, where we show that our method offers a significant enhancement in the dynamical data modeling capabilities of ESNs. Additionally, we also show that our method is orders of magnitude more computationally efficient compared to existing Gaussian process-based methods for dynamical data modeling, without compromises in the obtained predictive performance.**

***Index Terms*— Bayesian inference, Gaussian process, reservoir computing, sequential data modeling.**

I. Introduction

RECURRENT neural networks (RNNs) constitute a significant nonlinear approach for modeling dynamical systems as they entail recurrent connections between neurons, thus allowing for direct processing of temporal dependencies. RNNs possess the so-called universal approximation property [1], i.e., they are particularly capable of approximating arbitrary nonlinear dynamical systems with arbitrary precision [2], [3]. Additionally, they are also capable of reproducing temporal patterns similar to those they have been trained on. However, RNN training algorithms based on direct optimization of the network weights have led to less than satisfactory results [4], they usually exhibit slow convergence combined with high computational requirements, and often yield bifurcations and suboptimal estimates of the model parameters (local optima of the optimized objective functions). These issues can be largely attributed to the ill-posed nature of the RNN training problem, since parameter (weight) estimation involves inversion of a nonlinear dynamical system estimated from limited and noisy data [5].

A groundbreaking and surprisingly efficient network structure for RNNs that resolves the aforementioned issues was invented independently in the seminal works of Jaeger [6], who called these RNNs "echo state networks" (ESNs), and Maass *et al.* [7], who developed a similar approach for spiking neural networks and called the derived model the "liquid state machine" (LSM). These two innovative methodologies have given rise to the novel paradigm of reservoir computing (RC) [8], under which both the ESN and LSM network structures are usually subsumed. The RC paradigm avoids the shortcomings of typical gradient-descent-based RNN training by setting up the network structure in the following way [9].

1) An RNN is randomly created and remains unchanged during training. This RNN is called the *reservoir*. It is passively excited by the input signal and maintains in its state a nonlinear transformation of the input history.
2) The desired output signal is generated by a linear *readout* layer attached to the reservoir, which computes a linear combination of the neuron outputs from the input-excited reservoir *(reservoir states)*.

Hence, the function of the reservoir in RC networks can be compared to that of the kernel in kernel machine approaches [e.g., support vector machines (SVMs) [10], Gaussian processes GPs [11], relevance vector machines [12], and their variants] [13]. Input signals drive the nonlinear reservoir and produce a high-dimensional dynamical "echo response," which is used as a non-orthogonal basis to reconstruct the desired outputs. A schematic illustration of the RC approach is provided in Fig. 1.

As a result of these merits, RC networks have become very appealing with the computational intelligence community in the last years, being the epicenter of rigorous research efforts. For example, in [14] the authors have conducted an exhaustive experimental and theoretical analysis to investigate how simple a reservoir can become without compromising the model's performance, showing that a simple cycle reservoir topology is typically sufficient, with its memory capacity being arbitrarily close to the proved optimal value. In [15], an ESN network for modeling input/output signals defined in the complex domain has been proposed.

Among the existing RC implementations, most of the attention of the research community has been concentrated on the design of the network topologies and the selection of the neuron types. In this paper, we focus on ESNs, which usually employ analog neurons (typically linear), sigmoid or leaky-integrator [16] units, and simple sparsely connected graphs as their network topologies. An extensively studied subject in the field of ESN concerns the introduction of appropriate *goodness* measures of the reservoir structure. Indeed, the classical feature that reservoirs should possess is the echo state property. This property essentially states that the effect of a previous reservoir state and a previous input on a future state

Manuscript received August 19, 2010; revised July 2, 2011; accepted July 11, 2011. Date of publication July 29, 2011; date of current version August 31, 2011. This work was supported in part by the EU FP7 ALIZ-E project under Grant 248116.

The authors are with the Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2BT, U.K. (e-mail: soteri0s@me.com; y.demiris@imperial.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2011.2162109

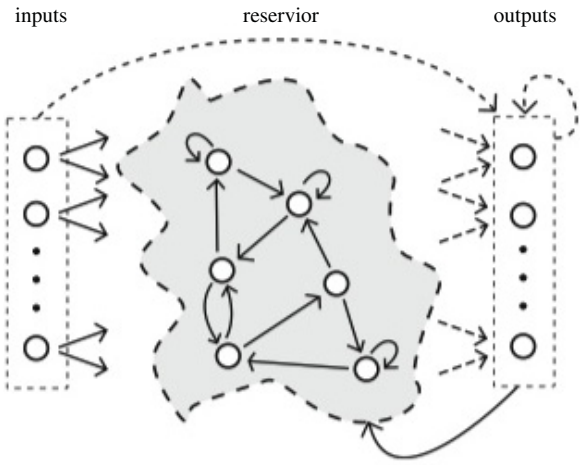

Fig. 1. Schematic overview of the RC approach [21].

should vanish gradually as time passes, and not persist or even get amplified. However, for most practical purposes, the echo state property can be easily satisfied by merely ensuring that the *reservoir weight matrix* $\boldsymbol{W}$ is *contractive*, i.e., by scaling the reservoir weight matrix so that its *spectral radius* $\rho(\boldsymbol{W})$ (that is, its largest absolute eigenvalue) is less than 1 [17]. Indeed, this condition has been proved to be sufficient in practical applications of ESNs, nevertheless, various researchers have also provided more rigorous global asymptotic stability conditions, providing better theoretical guarantees that ESNs will always perform well on a physical system (see [18]).

In this paper, we devise a novel Bayesian approach toward RC. We begin our analysis considering the imposition of a suitable prior distribution over the (trainable) weights of the synaptic connections between the reservoir neurons and the readout neurons of a postulated ESN (*readout weights*). Additionally, to endow our model with the capability of coping with observable datasets contaminated by noise and outliers, we introduce the fundamental assumption that the target values in the modeled populations comprise the superposition of some noiseless latent function of the reservoir state that the postulated model can learn, plus an independent white Gaussian noise signal. Based on this prior configuration, we eventually derive the predictive posterior distribution of the network readout, by effectively marginalizing out the readout weights. As we show, this construction eventually gives rise to a form of GPs with kernel functions that depend on the state values of an ESN reservoir employed to capture the dynamics within a set of sequentially dependent observations. We dub our nonparametric Bayesian approach toward reservoir computing the *echo state Gaussian process* (ESGP).

The ESGP combines both the merits of ESNs, in terms of their exceptional performance in modeling dynamical data, and GPs, in terms of their robustness to noise, their provision of a predictive distribution (instead of merely providing point predictions), and the simplicity of their training, which can be easily conducted by means of type-II maximum likelihood. As we shall explain next, inference in our model turns out to be extremely computationally efficient compared to other Gaussian-process-based methodologies for dynamical data modeling, the computational costs of which are orders of magnitude higher than the costs of our approach (e.g., [19]). Moreover, the provision of a full predictive posterior instead of single-point estimates gives rise to a dependable measure of uncertainty for each individual prediction, i.e., the model predictive variance at each considered test data point.

The efficacy of the proposed approach is evaluated on both synthetic applications, using well-known benchmark datasets, and real-life applications. Its performance is compared to both conventional ESN formulations using ridge-regression-based readout training, dynamical GP model formulations [19], and ESN-based SVM methodologies [20]. The remainder of this paper is organized as follows. In Section II, we provide a brief overview of the basic configuration of ESNs. In Section III, we concisely review the basic principles of GP regression. In Section IV, our proposed approach is introduced. In Section V, the experimental evaluation of our method is conducted. Finally, in the last section of this paper, our conclusions are drawn and our results are discussed.

II. ESNs

As already discussed, an ESN comprises two basic components, a discrete-time RNN, called the reservoir, and a linear readout output layer which maps the reservoir states to the actual output. Supervised ESN training is conducted by updating the reservoir state and network output as follows:

$$\begin{aligned}\boldsymbol{x}(t+1) = (1-\gamma)h(\boldsymbol{W}\boldsymbol{x}(t) + \ \boldsymbol{W}_{in}\boldsymbol{u}(t+1) \\ +\boldsymbol{W}_{out}\tilde{\boldsymbol{y}}(t)) + \gamma\,\boldsymbol{x}(t)\end{aligned} \tag{1}$$

$$\boldsymbol{y}(t+1) = \boldsymbol{W}_{readout}[\boldsymbol{x}(t+1); \boldsymbol{u}(t+1)] \tag{2}$$

where $\boldsymbol{x}(t)$ is the reservoir state at time t, $\boldsymbol{W}$ is the reservoir weight matrix, i.e., the matrix of the weights of the synaptic connections between the reservoir neurons, $\boldsymbol{u}(t)$ is the observed signal fed to the network at time t, $\tilde{\boldsymbol{y}}(t)$ is the desired value of the readout (i.e., the desired network output) at time t, $\boldsymbol{y}(t)$ is the obtained value of the readout at time t, $\gamma \geqslant 0$ is the *retainment rate* of the reservoir (with $\gamma > 0$ if leaky integrator neurons are considered), $\boldsymbol{W}_{readout}$ is the (linear) readout weights matrix, $\boldsymbol{W}_{in}$ and $\boldsymbol{W}_{out}$ are the weights of $\boldsymbol{u}(t)$ and $\boldsymbol{y}(t)$, and $h(\cdot)$ is the activation function of the reservoir. In the remainder of this paper, we will be considering hyperbolic-tangent reservoir neurons, i.e., $h(\cdot) \triangleq \tanh(\cdot)$.

After network training, the state update and output equations become

$$\begin{aligned}\boldsymbol{x}(t+1) = (1-\gamma)h(\boldsymbol{W}\boldsymbol{x}(t) + \boldsymbol{W}_{in}\boldsymbol{u}(t+1) \\ + \ \boldsymbol{W}_{out}\boldsymbol{y}(t)) + \gamma\,\boldsymbol{x}(t)\end{aligned} \tag{3}$$

$$\boldsymbol{y}(t+1) = \boldsymbol{W}_{readout}[\boldsymbol{x}(t+1); \boldsymbol{u}(t+1)]. \tag{4}$$

Given a training dataset $\{\boldsymbol{u}(t), \tilde{\boldsymbol{y}}(t)\}_{t=1}^{T}$, ESN training essentially comprises teacher-forced calculation of the corresponding reservoir states $\{\boldsymbol{x}(t)\}_{t=1}^{T}$ using (1), and application of a simple regression algorithm (e.g., linear regression or ridge regression) to train the readout weights $\boldsymbol{W}_{readout}$ on the resulting dataset $\{\boldsymbol{x}(t), \tilde{\boldsymbol{y}}(t)\}_{t=1}^{T}$ [9]. All the weight matrices to the reservoir ($\boldsymbol{W}$, $\boldsymbol{W}_{in}$, $\boldsymbol{W}_{out}$) are initialized randomly. The initial state of the reservoir is usually set to zero, i.e., $\boldsymbol{x}(0) = \boldsymbol{0}$.

III. GP Regression

GPs constitute one of the most important Bayesian machine learning approaches and are based on a particularly effective method for placing a prior distribution over the space of regression functions. They have a small number of tunable parameters, can be trained on relatively small training sets, and exhibit significant robustness to outliers and the ability to handle sparse data without getting prone to overtraining [11]. Compared to another popular form of discriminative kernel machines, i.e., the SVM [10], GPs possess several attractions, with the most significant being that the GP model produces an output with a clear probabilistic interpretation, providing a measure of uncertainty for the obtained predictions, contrary to SVMs which merely provide point predictions. As discussed in [22], this attractive property of GPs may result in better error-reject curves by taking into account the effect of the obtained uncertainty [11].

Let us consider an observation space $\mathcal{X}$. A GP $f(\boldsymbol{x})$, $\boldsymbol{x} \in \mathcal{X}$, is defined as *a collection of random variables, any finite number of which have a joint Gaussian distribution* [11]. A GP is completely specified by its mean function and covariance function. We define the mean function $m(\boldsymbol{x})$ and the covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ of a real process $f(\boldsymbol{x})$ as

$$\begin{aligned} m(\boldsymbol{x}) &= \mathbb{E}[f(\boldsymbol{x})] \\ k(\boldsymbol{x}, \boldsymbol{x}') &= \mathbb{E}\left[\left(f\left(\boldsymbol{x}\right) - m\left(\boldsymbol{x}\right)\right)\left(f\left(\boldsymbol{x}'\right) - m\left(\boldsymbol{x}'\right)\right)\right] \end{aligned} \tag{5}$$

and we will write the GP as

$$f(\boldsymbol{x}) \sim \mathcal{N}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x})). \tag{6}$$

Usually, for simplicity, and without any loss of generality, the mean of the process is taken to be zero, i.e., $m(\boldsymbol{x}) = 0$, although this is not necessary. Concerning selection of the covariance function, a large variety of kernel functions $k(\boldsymbol{x}, \boldsymbol{x}')$ might be employed, depending on the application considered [11]. This way, a postulated GP eventually takes the form

$$f(\boldsymbol{x}) \sim \mathcal{N}(0, k(\boldsymbol{x}, \boldsymbol{x})). \tag{7}$$

Let us suppose a set of independent and identically distributed samples $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) | i = 1, \ldots, N\}$, with the d-dimensional variables $\boldsymbol{x}_i$ being the observations related to a modeled phenomenon, and the scalars y_i being the associated target values. The goal of a regression model is, given a new observation $\boldsymbol{x}_*$, to predict the corresponding target value y_*, based on the information contained in the training set $\mathcal{D}$. The basic notion behind GP regression consists in the assumption that the observable (training) target values y in a considered regression problem can be expressed as the superposition of a GP over the input space $\mathcal{X}$, $f(\boldsymbol{x})$, and an independent white Gaussian noise

$$y = f(\boldsymbol{x}) + \epsilon \tag{8}$$

where $f(\boldsymbol{x})$ is given by (7), and

$$\epsilon \sim \mathcal{N}(0, \sigma^2). \tag{9}$$

Under this condition, the joint normality of the training target values $\boldsymbol{y} = [y_i]_{i=1}^N$ and some unknown target value y_*, approximated by the value f_* of the postulated GP evaluated at the observation point $\boldsymbol{x}_*$, yields [11]

$$\begin{bmatrix} \boldsymbol{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K}(X, X) + \sigma^2 \boldsymbol{I}_N & \boldsymbol{k}(\boldsymbol{x}_*) \\ \boldsymbol{k}(\boldsymbol{x}_*)^{\mathrm{T}} & k(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix}\right) \tag{10}$$

where

$$\boldsymbol{k}(\boldsymbol{x}_*) \triangleq [k(\boldsymbol{x}_1, \boldsymbol{x}_*), ..., k(\boldsymbol{x}_N, \boldsymbol{x}_*)]^{\mathrm{T}} \tag{11}$$

$X = \{\boldsymbol{x}_i\}_{i=1}^N$, $\boldsymbol{I}_N$ is the $N \times N$ identity matrix, and $\boldsymbol{K}$ is the matrix of the covariances between the N training data points *(design matrix)*

$$\boldsymbol{K}(X, X) \triangleq \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) \ldots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) \ldots & k(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & k(\boldsymbol{x}_N, \boldsymbol{x}_2) \ldots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}. \tag{12}$$

Then, from (10), and conditioning on the available training samples, we can derive the expression of the model predictive distribution, yielding

$$p(f_* | \boldsymbol{x}_*, \mathcal{D}) = \mathcal{N}(f_* | \mu_*, \sigma_*^2) \tag{13}$$

where

$$\mu_* = \boldsymbol{k}(\boldsymbol{x}_*)^{\mathrm{T}} \left(\boldsymbol{K}(X, X) + \sigma^2 \boldsymbol{I}_N\right)^{-1} \boldsymbol{y} \tag{14}$$

and

$$\sigma_*^2 = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}(\boldsymbol{x}_*)^{\mathrm{T}} \left(\boldsymbol{K}(X, X) + \sigma^2 \boldsymbol{I}_N\right)^{-1} \boldsymbol{k}(\boldsymbol{x}_*). \tag{15}$$

It is interesting to observe that the predictive variance of the GP model, given by (15), does not depend on the training target values, but only on the training input values. Indeed, the predictive variance is the difference between two terms: the first term $k(\boldsymbol{x}_*, \boldsymbol{x}_*)$ is simply the prior covariance, from which a (positive) term is subtracted, representing the information the observations give us about the regression function.

Regarding optimization of the hyperparameters of the employed covariance function (kernel), say $\boldsymbol{\theta}$, and the noise variance σ^2 of a GP model, this is usually conducted by type-II maximum likelihood, i.e., by maximization of the model marginal likelihood (evidence). Using (10), it is easy to show that the evidence of the GP regression model yields

$$\begin{aligned} \log p(\boldsymbol{y} | X; \boldsymbol{\theta}, \sigma^2) = &-\frac{N}{2} \log 2\pi - \frac{1}{2} \log \left|\boldsymbol{K}(X, X) + \sigma^2 \boldsymbol{I}_N\right| \\ &-\frac{1}{2} \boldsymbol{y}^{\mathrm{T}} \left(\boldsymbol{K}(X, X) + \sigma^2 \boldsymbol{I}_N\right)^{-1} \boldsymbol{y}. \end{aligned} \tag{16}$$

IV. Proposed Approach

A. Model Formulation

Let us again consider the readout expression of an ESN comprising K reservoir neurons. Let the network output $\boldsymbol{y}(t)$ consist of M component responses, i.e. $\boldsymbol{y}(t) = \left[y_j(t)\right]_{j=1}^M$. Then, from (2) we have

$$y_j(t) = \boldsymbol{w}_j^{\mathrm{T}} \boldsymbol{\psi}(t) \tag{17}$$

where

$$\boldsymbol{\psi}(t) \triangleq [\boldsymbol{x}(t); \boldsymbol{u}(t)]. \tag{18}$$

Let us now impose a spherical Gaussian prior over the readout weights $\boldsymbol{w}_j$, such that

$$\boldsymbol{w}_j \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{19}$$

Under this setting, we have for the mean and covariance of the readout component responses $y_j(t)$

$$\mathbb{E}\left[y_j(t)\right] = \mathbb{E}\left[\boldsymbol{w}_j^{\mathrm{T}}\right]\boldsymbol{\psi}(t) = 0 \tag{20}$$

and

$$\mathbb{E}\left[y_j(t_1)y_j(t_2)\right] = \boldsymbol{\psi}(t_1)^{\mathrm{T}}\mathbb{E}\left[\boldsymbol{w}_j\boldsymbol{w}_j^{\mathrm{T}}\right]\boldsymbol{\psi}(t_2) = \boldsymbol{\psi}(t_1)^{\mathrm{T}}\boldsymbol{\psi}(t_2). \tag{21}$$

Thus, under the Gaussian prior assumption (19), it turns out that $y_j(t_1)$ and $y_j(t_2)$ are jointly Gaussian with zero mean and covariance given by the dot product $\boldsymbol{\psi}(t_1)^{\mathrm{T}}\boldsymbol{\psi}(t_2)$, for any $j \in \{1, \ldots, M\}$, and $\forall t_1, t_2$. In other words, under our Bayesian approach, the distributions of the ESN readouts turn out to yield a GP of the form

$$\left[y_j(t)\right]_{t=t_1}^{t_T} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_r(\boldsymbol{\Psi}, \boldsymbol{\Psi})) \tag{22}$$

where $\boldsymbol{\Psi}$ is the *design matrix*

$$\boldsymbol{\Psi} = [\boldsymbol{\psi}(t_1), \ldots, \boldsymbol{\psi}(t_T)] \tag{23}$$

and $\boldsymbol{K}_r$ is given by

$$\boldsymbol{K}_r(\boldsymbol{\Psi}, \boldsymbol{\Psi}) \triangleq \begin{bmatrix} k_r(\boldsymbol{\psi}(t_1), \boldsymbol{\psi}(t_1)) & \ldots & k_r(\boldsymbol{\psi}(t_1), \boldsymbol{\psi}(t_T)) \\ \vdots & \vdots & \vdots \\ k_r(\boldsymbol{\psi}(t_T), \boldsymbol{\psi}(t_1)) & \ldots & k_r(\boldsymbol{\psi}(t_T), \boldsymbol{\psi}(t_T)) \end{bmatrix} \tag{24}$$

with the kernels of the obtained GPs being functions of the reservoir state vectors, in the form

$$k_r(\boldsymbol{\psi}(t_1), \boldsymbol{\psi}(t_2)) \triangleq \boldsymbol{\psi}(t_1)^{\mathrm{T}}\boldsymbol{\psi}(t_2). \tag{25}$$

Generalizing the above results to allow for the utilization of kernels of any kind, in this paper we introduce a novel Bayesian treatment of ESNs, namely, the ESGP. For example, in case a Gaussian radial basis function (RBF) kernel is considered, the definition of the ESGP model yields a prior distribution of the form (22) with its kernel function (reservoir kernel) given by

$$k_r(\boldsymbol{\psi}(t_1), \boldsymbol{\psi}(t_2)) \triangleq \exp\left[-\frac{||\boldsymbol{\psi}(t_1) - \boldsymbol{\psi}(t_2)||^2}{2\lambda^2}\right]. \tag{26}$$

Definition 1: We define as the ESGP a GP the covariance function of which is taken as a kernel function over the states of an ESN reservoir postulated to capture the dynamics within a set of sequentially interdependent observations.

An interesting feature of the ESGP model is that, despite the clear analogy between the computation of the high-dimensional projection performed by a reservoir and by kernel methods, in the case of the ESGP model feature mapping computation is explicit (i.e., no kernel trick is used, but the high-dimensional feature projections are explicitly computed, being the reservoir states), and the feature mapping has an internal memory, due to its being a dynamical system.

B. Model Inference

Let us consider an ESGP with K reservoir neurons and M readout signals. To endow our model with increased robustness to observation noise, we additionally assume that the target signals of an observed phenomenon modeled using the postulated ESGP model consist of a latent function of the input signals, which is learnable by the considered ESGP model, superimposed on an independent white Gaussian noise signal, that is, we adopt the hypothesis that the available training target signals $\tilde{y}_j$ are given by

$$\tilde{y}_j(t) = y_j(t) + \epsilon \tag{27}$$

where the $y_j(t)$ are distributed as in (22), and the distribution of the noise ϵ is given by (9). Then, following the analysis of Section III, the predictive density of the postulated ESGP model at a test time point t_* yields

$$p(y_{j*}|\boldsymbol{\psi}(t_*), \mathcal{D}) = \mathcal{N}(y_{j*}|\mu_{j*}, \sigma_*^2) \tag{28}$$

where

$$\mu_{j*} = \boldsymbol{k}_r(\boldsymbol{\psi}(t_*))^{\mathrm{T}}\left(\boldsymbol{K}_r(\boldsymbol{\Psi}, \boldsymbol{\Psi}) + \sigma^2\boldsymbol{I}_T\right)^{-1}\tilde{\boldsymbol{y}}_j \tag{29}$$

$$\sigma_*^2 = k_r(\boldsymbol{\psi}(t_*), \boldsymbol{\psi}(t_*)) - \boldsymbol{k}_r(\boldsymbol{\psi}(t_*))^{\mathrm{T}}\left(\boldsymbol{K}_r(\boldsymbol{\Psi}, \boldsymbol{\Psi}) + \sigma^2\boldsymbol{I}_T\right)^{-1}\boldsymbol{k}_r(\boldsymbol{\psi}(t_*)) \tag{30}$$

$$\boldsymbol{k}_r(\boldsymbol{\psi}(t_*)) \triangleq [k_r(\boldsymbol{\psi}(t_1), \boldsymbol{\psi}(t_*)), \ldots, k_r(\boldsymbol{\psi}(t_T), \boldsymbol{\psi}(t_*))]^{\mathrm{T}} \tag{31}$$

and $\tilde{\boldsymbol{y}}_j = [\tilde{y}_j(t_\tau)]_{\tau=1}^T$. Similar, the model evidence will be given by

$$\log p(\tilde{\boldsymbol{y}}_j|\boldsymbol{\Psi}; \sigma^2) = -\frac{T}{2}\log 2\pi - \frac{1}{2}\log\left|\boldsymbol{K}_r(\boldsymbol{\Psi}, \boldsymbol{\Psi}) + \sigma^2\boldsymbol{I}_T\right| - \frac{1}{2}\tilde{\boldsymbol{y}}_j^{\mathrm{T}}\left(\boldsymbol{K}_r(\boldsymbol{\Psi}, \boldsymbol{\Psi}) + \sigma^2\boldsymbol{I}_T\right)^{-1}\tilde{\boldsymbol{y}}_j. \tag{32}$$

Estimation of the model hyperparameters, i.e., the noise hyperparameter σ^2 as well as of the hyperparameters of the employed (reservoir) kernels, is conducted by optimization of the model evidence (32). To perform this optimization task, we here employ the scaled conjugate gradient descent [23] algorithm.

C. Relations to Existing ESN Treatments

As we shall show in the following, the ESGP model generalizes conventional ESNs, treated by means of ridge or linear regression, under a Bayesian perspective, and includes them as special subcases. Indeed, if we consider a simple linear kernel for our model of the form (25), then the resulting expression of $\boldsymbol{K}_r(\boldsymbol{\Psi}, \boldsymbol{\Psi})$ turns out to be essentially low-rank by construction. Expanding (29) and (30) in terms of $\boldsymbol{K}_r(\boldsymbol{\Psi}, \boldsymbol{\Psi})$ and $\boldsymbol{\psi}(t)$, and using the matrix inversion lemma, the expressions of the ESGP predictive mean and variance can be restated in the form

$$\mu_{j*} = \boldsymbol{\psi}(t_*)^{\mathrm{T}}\boldsymbol{A}^{-1}\boldsymbol{\Psi}\tilde{\boldsymbol{y}}_j \tag{33}$$

and

$$\sigma_*^2 = \sigma^2\boldsymbol{\psi}(t_*)^{\mathrm{T}}\boldsymbol{A}^{-1}\boldsymbol{\psi}(t_*) \tag{34}$$

where

$$\boldsymbol{A} = \boldsymbol{\Psi}\boldsymbol{\Psi}^{\mathrm{T}} + \sigma^2\boldsymbol{I}_{K_d} \tag{35}$$

and $K_d = K + d$, with d being the dimensionality of $\boldsymbol{u}(t)$.

Examining the reduced expression (33) of the model predictive mean, we observe that this expression is essentially identical to the expression of the predictions generated by ESNs trained by means of ridge regression [24]. In other words, in the special case that we consider a simple linear kernel for our model, the predictions generated by the ESGP coincide with those generated by ESNs trained by means of ridge regression. Additionally to that though, the ESGP does also provide a measure of uncertainty regarding the generated predictions (i.e., the computed predictive variances), while it also allows the estimation of its noise hyperparameter σ^2 (i.e., the regularization parameter in ridge regression-based ESNs) by means of type-II maximum likelihood, hence obviating the need of performing cross validation, which is clearly more tedious in terms of computational requirements. Finally, from these results we also deduce that, by explicitly setting $\sigma^2 = 0$ and again considering a linear kernel for the ESGP model, the ESGP-generated predictions coincide with those of ESNs trained by means of plain linear regression.

V. Experiments

In the following section, we provide a thorough experimental evaluation of the ESGP model, considering both classical benchmark tasks and real-world applications. In our experimental evaluations, we consider reservoirs comprising analog neurons, with tanh transfer functions. To demonstrate the advantages of our approach, we also evaluate linear-regression-based ESNs, ridge regression-based ESNs, and support vector echo state machine (SVESM) models with ϵ-insensitive loss functions [20], *using the same reservoirs* as the evaluated ESGP models, as well as the dynamic GP (DynGP) method of [19], trained by means of a two-stage maximum *a posteriori* (MAP) estimation method. The reservoir parameters of the ESN-based methods were selected so as to maximize the performance of the considered baseline method, i.e., the linear-regression-based ESN. The ESGP and SVESM methods were evaluated considering Gaussian RBF kernels. Regarding the DynGP method, we note that testing was conducted by presenting to the algorithm the first δt samples of the test sequences, where δt is the reservoir warm-up time employed by the ESN-based methods.

Our source codes were developed in MATLAB, and made partial use of the RC Toolbox [8], as well as of software provided by Neil Lawrence [25]. Our implementation of the SVESM method was based on the library of support vector machine of [26], written in C, hence providing a much more computationally efficient implementation compared to the rest of the evaluated methods. Therefore, the observed execution times of the evaluated algorithms are not fully comparable. Our experiments were executed on a Macintosh platform with 4 GB RAM, and an Intel Core 2 Duo processor, running Mac OS X 10.6. In Table I, we summarize the configuration details of the employed reservoirs in the considered experiments.

In our experiments, the weights of the input $\boldsymbol{u}(t)$, stored in the matrix $\boldsymbol{W}_{in}$, as well as the nonzero elements of the reservoir weights matrix $\boldsymbol{W}$, are drawn from the standard Gaussian distribution. The results provided in the remainder of this section are averages over 10 different random reservoir initializations (common for all the evaluated RC-based methods). Finally, ESN training was conducted using fivefold cross validation.

A. Mackey–Glass Series

The Mackey–Glass delay differential equation has provided one of the most classical benchmark tasks for time series modeling. In a discrete time setting, the Mackey–Glass delay differential equation is approximated as

$$y(t+1) = y(t) + \delta\left(0.2\frac{y(t-\tau/\delta)}{1+y(t-\tau/\delta)^{10}} - 0.1y(t)\right) \quad (36)$$

with the stepsize δ typically set to $\delta = 1/10$ [6], [17]. The resulting time series is afterward rescaled into the range $[-1, 1]$ by application of a tangent-hyperbolic transform $y_{ESN}(t) = \tanh(y(t) - 1)$, so that it can be used to train ESNs with tanh activation functions in the reservoir. The system is chaotic for values of the delay time $\tau < 16.8$.

In our experiment, training sequences were generated from (36) for delay time $\tau = 17$, similar to [6]. Initially, the ESN-based models were trained using a signal comprising 6000 time points, and initial transient was washed out by employing a reservoir warm-up time of 100 steps. Subsequently, evaluation was conducted by simulating the trained models *using new time series* of 250 samples, with a reservoir warm-up time of 100 time steps. Using the simulated network outputs, we compare the performance of the considered models by calculating the obtained normalized absolute error (NAE) on a specific prediction horizon. The NAE on a t-step prediction horizon reads

$$\mathrm{NAE}_t = \sqrt{\frac{1}{s^2}\left(y(\mathrm{warmup}+t) - \tilde{y}(\mathrm{warmup}+t)\right)^2} \quad (37)$$

where s^2 is the empirical variance of the target signal. Prediction on a t-step horizon in all the evaluated models was conducted by iteratively applying the predictor t times in a "generative" mode, where on each step it takes its own last prediction to do the next prediction.

Here, we consider a commonplace selection for the Mackey–Glass system prediction horizon, i.e., prediction 84 and 120 steps after the washout time elapses [6]. The obtained results are provided in Table II and Fig. 2. These results are means and standard deviations of the obtained performance metrics over 50 test sequences. We observe that the ESGP with Gaussian RBF kernel performs much better than the considered alternatives, being capable of obtaining much lower NAE_{84}, NAE_{120}, and $\langle \mathrm{NAE}_t \rangle$ (mean NAE_t) values. We also note that the performance of the SVESM is worse than the performance of the rest of the considered ESN-based methods.

Regarding the performance of the DynGP method, we would like to make two substantial comments. The first one concerns the model predictive performance. As we notice, the DynGP is the worst performing method in this application. Our second comment regards the overwhelming computational

TABLE I

CONFIGURATION OF THE EMPLOYED RESERVOIRS IN OUR EXPERIMENTS

Parameter	Mackey–Glass	Henon map	Figure 8	Learning to grasp	Sequential data classification	Human motion modeling
Reservoir neurons	400	100	1000	20	5	100
Spectral radius	0.99	0.90	0.98	0.98	0.998	0.998
Reservoir connectivity	0.1	0.1	0.2	0.2	0.1	0.1
Warm-up time (model training)	100	100	0	0	0	0
Warm-up time (model evaluation)	100	100	1	3	1	10
γ	0.8	0.8	0	0.98	0	0.998

TABLE II

MACKEY–GLASS SERIES: PERFORMANCE OF THE EVALUATED MODELS

Model	Linear-regression-based ESN	Ridge-regression-based ESN	ESGP	SVESM	DynGP
NAE_{84}	0.0166 (0.0102)	0.0119 (0.0076)	0.0017 (0.00013)	0.0657 (0.0078)	0.1116 (0.0309)
NAE_{120}	0.0280 (0.0150)	0.0175 (0.0099)	0.0047 (0.00031)	0.0756 (0.0198)	0.7457 (0.1817)
$\langle NAE_t \rangle$	0.0138	0.0094	0.0016	0.0745	0.5396

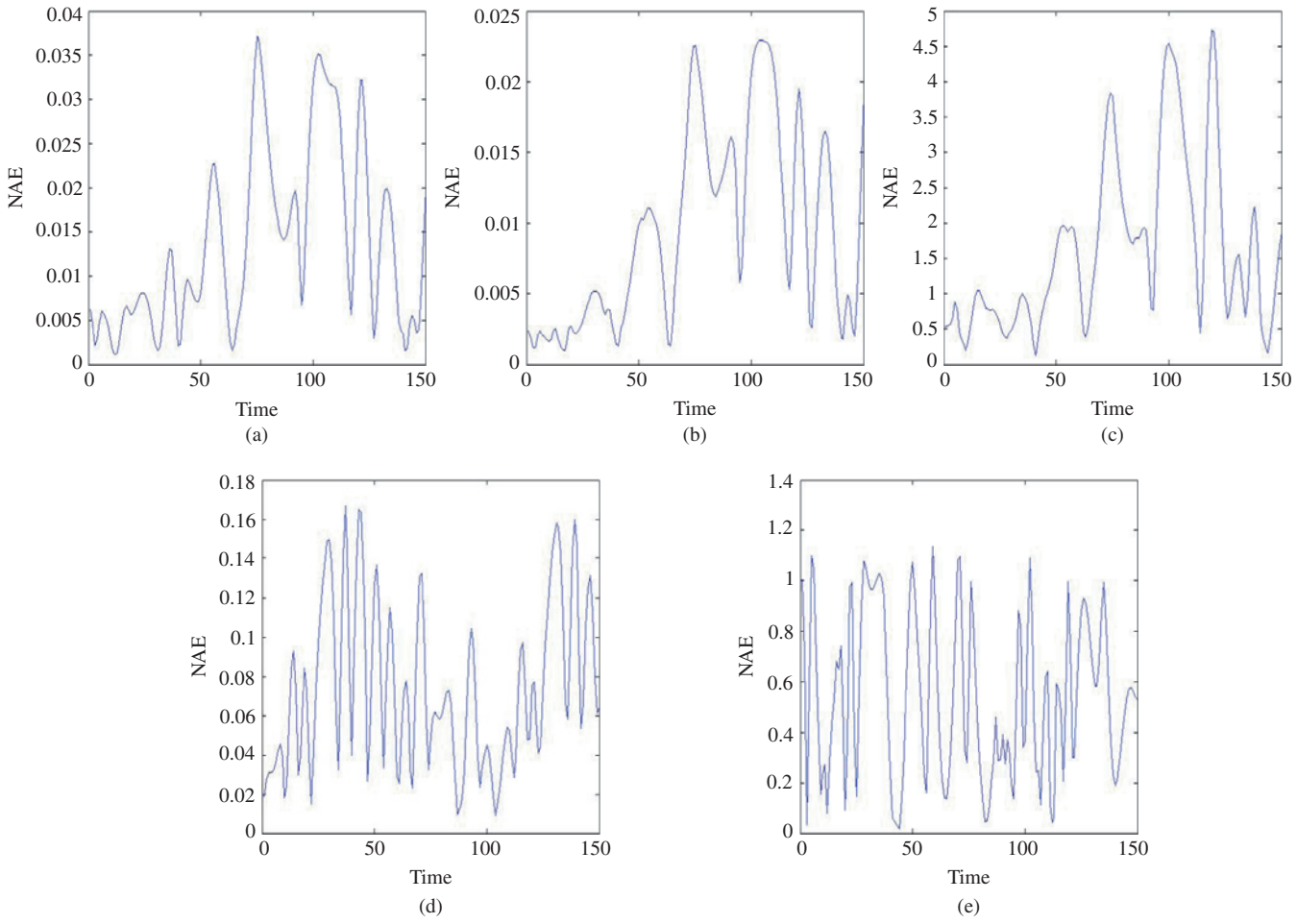

Fig. 2. Mackey–Glass series: NAE values over the 150 prediction time points of the evaluated models. (a) Linear regression ESN. (b) Ridge regression ESN. (c) ESGP. (d) SVESM. (e) DynGP.

costs of the DynGP method. Indeed, whereas running the ESGP method required a maximum execution time of roughly 50 s, the DynGP method required more than 7×10^3 s, i.e., around 2 h, to execute only 15 repetitions of the estimation algorithm. Moreover, when we attempted to run the algorithm using the convergence criteria suggested by the authors of [19], we noticed that the algorithm kept running for more than 2 days, without even training having converged (around 330 repetitions had been performed). In other words, we observe that the computational costs of the DynGP method,

TABLE III
HENON MAP: PERFORMANCE OF THE EVALUATED MODELS

Model	Linear-regression-based ESN	Ridge-regression-based ESN	ESGP	SVESM	DynGP
NAE_{84}	0.6439	0.6020	0.5745	0.6469	0.0921
NAE_{120}	0.9304	0.9260	0.7512	1.3939	0.7549
NAE_{400}	0.6995	0.6641	0.5641	0.5156	0.4912
$\langle \text{NAE}_t \rangle$	0.9172	0.9083	0.8883	0.9734	0.9121

combined with its poor predictive performance, make it clearly inapplicable to the Mackey–Glass series generation problem. Finally, to provide a better insight into how the proposed ESGP method compares with its alternatives in regard to the imposed computational burden, we mention that under our unoptimized MATLAB implementation, the ridge regression-based ESN required roughly 35 s. This figure compared to the approximately 50 s required by the ESGP method shows that the ESGP offers a very good tradeoff between computational complexity and sequential data modeling performance.

B. Henon Map

Here, we consider another typical benchmark in the field of RNNs, i.e., the Henon map chaotic process [27]. Henon map is a discrete-time dynamical system exhibiting a characteristic chaotic behavior. It receives as input a 2-D point $\boldsymbol{y}(t) = [y_1(t), y_2(t)]$, and maps it to a new point $\boldsymbol{y}(t+1) = [y_1(t+1), y_2(t+1)]$ on the 2-D plane, given by

$$y_1(t+1) = y_2(t) + 1 - \alpha y_1^2(t) \tag{38}$$

$$y_2(t+1) = \beta y_1(t) \tag{39}$$

where $\alpha = 1.4$ and $\beta = 0.3$. The starting point of the Henon map considered here is $\boldsymbol{y}(0) = \boldsymbol{0}$.

In our experiments, the evaluated ESN-based models were trained using the first 1000 samples of the Henon map, initial transient was washed out by employing a reservoir warm-up time of 100 steps. Subsequently, evaluation was conducted by using the trained models to generate *the next 2000 samples of the Henon map*, with the employed reservoirs being teacher-driven for the first 100 time points. The performance of the evaluated models in terms of the obtained NAE_{84}, NAE_{120}, NAE_{400}, and $\langle \text{NAE}_t \rangle$ metrics is depicted in Table III. As we observe, the ESGP model performs better than the considered alternatives. It is noteworthy that the SVESM is the worst performing method in this experiment. Regarding the DynGP method, we observe that, even though the DynGP offers better NAE_{84}, NAE_{120}, and NAE_{400} metrics than the competition, the obtained mean performance ($\langle \text{NAE}_t \rangle$ metric) of the DynGP model does not perform as well. This is basically due the very bad prediction performance of the DynGP method at some time points, which adversely affects the average method performance.

C. Non-Smooth Figure 8

In this experiment, we evaluate the effectiveness of our model in learning complex sequential patterns. For this purpose, we consider a non-smooth figure 8, with the points of the figure moving around very quickly, and each cycle comprising only few points. To obtain this signal, we use the function dataset_figure8 the Reservoir Computing Toolbox [8], which generates the figure 8 as a superposition of a sine on the horizontal direction, and a cosine of half the sine's frequency on the vertical direction.

The evaluated models were trained using a sequence of 200 data points from the figure 8 trajectory, and no reservoir warmup was employed. On the sequel, the trained models were evaluated over 1000 time steps, with the reservoir being warmed up only for one time step. In Fig. 3, we provide the trajectories produced by the evaluated methods. As we observe, the ESGP with Gaussian RBF kernel works considerably better than the SVESM and the linear-regression-based ESN, and slightly better than the ridge-regression-based ESN. Specifically the ridge-regression-based ESN yields a normalized root mean square error (NRMSE) equal to 1.2961, whereas the ESGP with Gaussian RBF kernel yields an NRMSE equal to 0.8144.

D. Learning to Grasp Stationary Objects

In this experiment, we consider a real-life application in the field of robotics: the aim is to teach by demonstration a robot how to grasp a stationary object under different settings. The five different experimental cases considered here, adopted from [28], are depicted in Fig. 4. To conduct our experiments, we have made use of the *iCub platform* [Fig. 5(a)], a humanoid robot developed by the RobotCub Consortium [29]. In the model training phase, five different human demonstrators were asked to perform each one of the five tasks considered here, with the iCub observing their actions. It is significant to note that the human subjects were not asked to try to follow a strictly defined trajectory (e.g., a trajectory as straight as possible), but rather perform their movements in a way as natural to them as possible. This way, the available training datasets are both limited, since the obtained trajectories were of variable length between 20 and 50 samples in each case, as well as considerably noisy, thus providing a clearly realistic task learning scenario.

The pair of stereo cameras on board the iCub platform were used to capture the demonstrated information, with the camera frame rate set to 20 Hz, and the resolution being equal to 320×240 pixels as illustrated, e.g., in Fig. 5(b) and (c). Markers were placed on human subjects [Fig. 5(d)] to track the points of interest. Based on this setting, the positions of the tracked markers on the 3-D space were presented to the trained models, with the goal to learn what trajectory to follow in order to reach the objects of interest under the five considered alternative scenarios. At the testing phase of our experiment, another nine subjects were asked to perform the same five

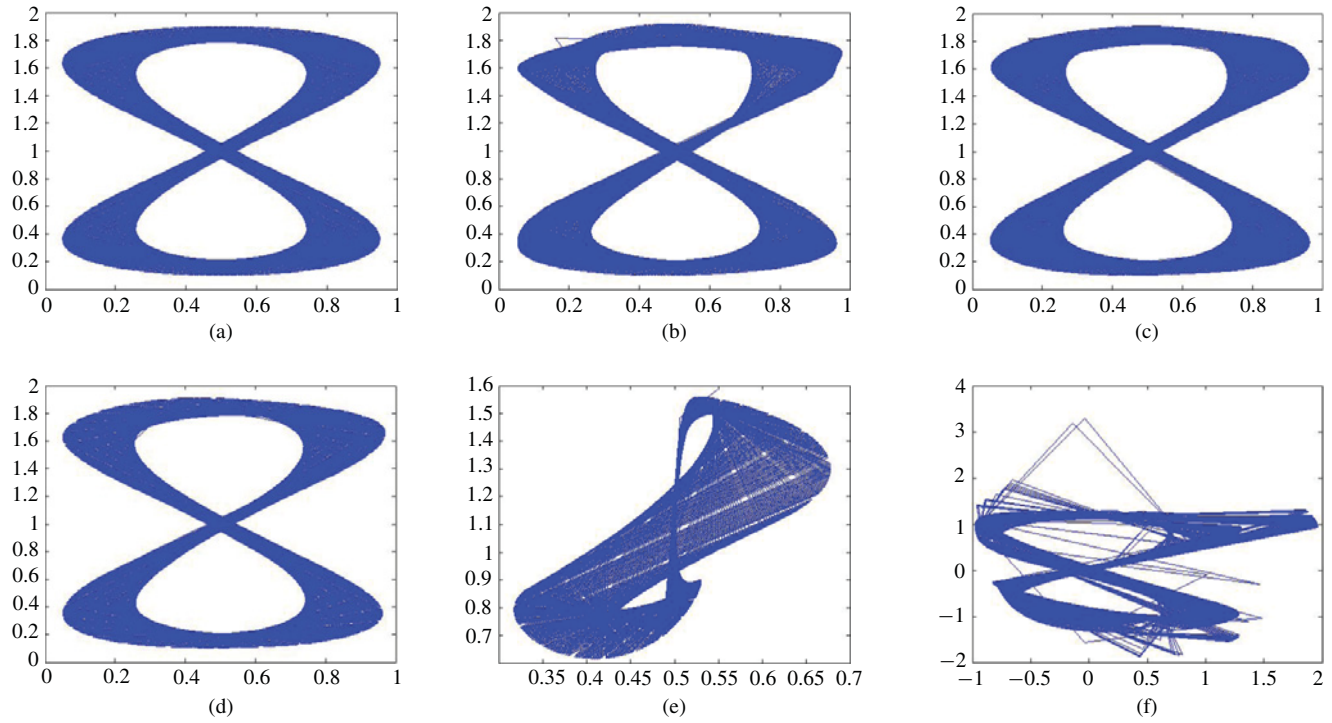

Fig. 3. Non-smooth figure 8. (a) Original time series. (b) Reconstruction obtained by the ESN trained using linear regression. (c) Reconstruction obtained by the ESN trained using ridge regression. (d) Reconstruction obtained by the ESGP. (e) Reconstruction obtained by the SVESM. (f) Reconstruction obtained by the DynGP method.

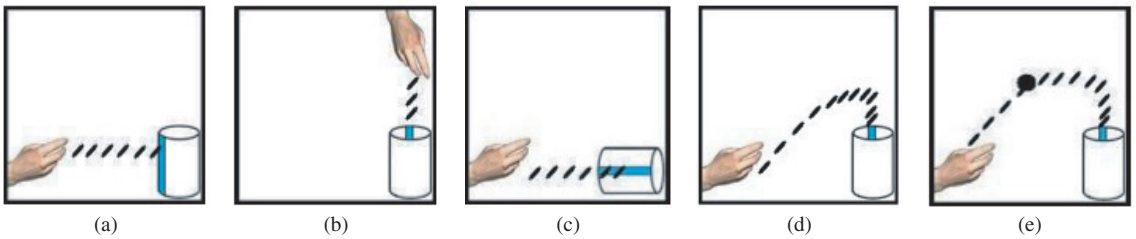

Fig. 4. Learning to grasp stationary objects. Graphical illustration of the considered experimental cases. Hand positions in the diagrams indicate the starting points of the experiments, while the taught paths are denoted by black slashes. All subjects were requested to use their inferior arm (left arms in all cases), and keep their forearms orthogonal to the blue strips while approaching the objects. The black patch in (e) denotes the waypoint area the subjects have to navigate their arms through. (a) First experimental case. (b) Second experimental case. (c) Third experimental case. (d) Fourth experimental case. (e) Fifth experimental case.

tasks. The iCub was allowed to observe only the first three samples of the trajectories followed by the human subjects, and subsequently, the trained models were asked to generate the rest of the trajectories. In other words, the trained models were evaluated in terms of their capacity to autonomously generate the trajectories the human subjects intended to follow in order to execute the given tasks, based on the knowledge obtained in the model training phase, and given the few first points of the trajectories followed by the test subjects.

In Table IV, we provide the obtained root mean square error (RMSE) of the evaluated methods for each task. We also compare with the performance obtained by the template-based path imitation method proposed in [28][1]. As we observe, the ESGP outperforms the method of [28], which is specifically designed for the purpose of path imitation by robotic systems, in four out of the five considered tasks. It also performs much better than the linear-regression-based and ridge-regression-based ESNs. We also observe that, while the SVESM performs better than the linear-regression-based and ridge-regression-based ESNs in the first four tasks, it turns out to be the worst performing ESN-based method in the case of the fifth task. Finally, note how poorly the DynGP method performs in this application. We tend to attribute this poor outcome of the DynGP to the significant amounts of noise and outliers in the considered training datasets, and the significant variability in the training and test sequences, combined with the limited availability of training data. However, trying to substantiate this claim is clearly beyond the scope of this paper.

[1]Note, though, that the method of [28] is basically designed for performing transfer learning between tasks, a capability that the rest of the methods do not possess. Therefore, in these experiments we do not utilize the full potential of the method of [28], to ensure fair comparison.

E. Sequential Data Classification

In this experiment, we show how the predictive distributions provided by the ESGP model can be used to measure the

TABLE IV
LEARNING TO GRASP STATIONARY OBJECTS: AVERAGE OBTAINED RMSES FOR EACH TASK

Method	Linear-regression-based ESN	Ridge-regression-based ESN	ESGP	SVESM	Template-based method of [28]	DynGP
Task #1	2.69×10^3	38.04	35.38	32.52	24.6	195.81
Task #2	1.072×10^3	30.23	10.30	16.37	10.7	4.67×10^3
Task #3	337.97	34.14	23.78	23.97	41.2	1.13×10^3
Task #4	361.65	90.62	24.54	24.90	38.6	422.69
Task #5	27.65	26.86	25.94	36.46	27.0	1.06×10^3

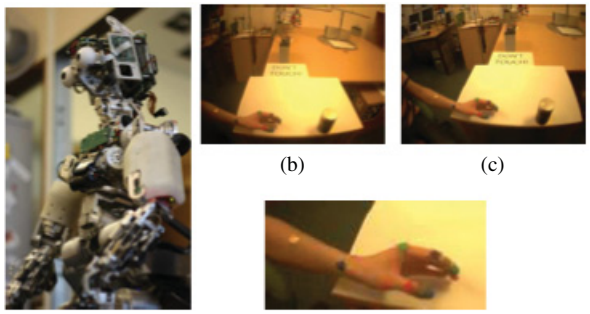

Fig. 5. Learning to grasp stationary objects. (a) Robotic platform used in our experiments. (b) and (c) human subject with markers captured by the left and right cameras of the iCub, respectively. (d) First experimental case: Marker locations placed on the left arm of the human subjects.

affinity between sequentially evolving patterns, and how we can exploit this information to effect sequential data classification. Subsequently, we compare the performance of our approach with the most commonly used method for sequential data classification, i.e., hidden Markov models (HMMs) with Gaussian or Student's-t mixtures as their observation emission models.

As we have already discussed, the ESGP provides a predictive distribution which constitutes a function of the state value of its employed reservoir. Let us consider two ESGP models using exactly the same reservoir: a model $\mathcal{L}_1$ with predictive density

$$p(\boldsymbol{y}_*|\mathcal{L}_1) = \prod_{j=1}^{M} \mathcal{N}(y_{j*}|\tilde{\mu}_{j*}, \tilde{\sigma}_j^2) \tag{40}$$

trained to model (and generate) a specific class of sequential data, and a model $\mathcal{L}_0$ with predictive density

$$p(\boldsymbol{y}_*|\mathcal{L}_0) = \prod_{j=1}^{M} \mathcal{N}(y_{j*}|\hat{\mu}_{j*}, \hat{\sigma}_j^2) \tag{41}$$

trained on a single sequential data observation (one given sequence) of the same class as the one modeled by $\mathcal{L}_1$. As is easy to perceive, if the above trained models are really capable of modeling and correctly generating the considered patterns, then using an appropriate statistical measure of dissimilitude between the two models $d(\mathcal{L}_0, \mathcal{L}_1)$ would be expected to yield a very low distance value between the two models. On the other hand, the same distance metric calculated between model $\mathcal{L}_0$ and an ESGP model $\mathcal{L}_2$ employing the same reservoir but trained to generate sequences pertaining to a different class should be yielding a much higher distance value.

To examine whether this assumption does really hold, and, hence, whether the ESGP can be successfully applied to sequential data classification under such a model distance-based setting, we consider a text-dependent speaker identification task. Our experiments are based on the Japanese Vowels Dataset [30] from the UCI machine learning repository [31]. In this dataset, the passphrase used for speaker identification purposes comprises two Japanese vowels, /ae/, successively uttered by nine male speakers. For each utterance, a 12-degree linear prediction analysis is applied to obtain a discrete-time series with 12 linear prediction coding cepstrum coefficients. There is a set of 270 time series for model training, and a set of 370 time series for testing.

Our experimental setup was the following: Initially, for each modeled speaker, we trained a different ESGP model as the speaker model, using the available training data. Subsequently, for each test sequence we trained an individual ESGP, employing the same reservoir with the one used by the trained speaker models (comprising simple tanh neurons). Finally, for each test sequence, we calculated the distance between its ESGP model and each one of the trained speaker models, $d(\mathcal{L}_0, \mathcal{L}_i)$. Using this information, each test sequence was eventually attributed to the speaker yielding the lowest dissimilitude $d(\mathcal{L}_0, \mathcal{L}_i)$. In this paper, the dissimilitude $d(\mathcal{L}_0, \mathcal{L}_1)$ between a test sequence model $\mathcal{L}_o$ and a class (speaker) model $\mathcal{L}_1$ is measured by means of the Kullback–Leibler divergence

$$\begin{aligned} \mathrm{KL}(\mathcal{L}_0||\mathcal{L}_1) \triangleq & \\ & \sum_{t_*=1}^{T} \left\langle \log \mathcal{N}\left(\boldsymbol{y}(t_*)|\{\hat{\mu}_{j*}\}_{j=1}^{M}, \hat{\sigma}_*^2\right) \right\rangle_{\mathcal{N}\left(\boldsymbol{y}(t_*)|\{\hat{\mu}_{j*}\}_{j=1}^{M}, \hat{\sigma}_*^2\right)} \\ & - \sum_{t_*=1}^{T} \left\langle \log \mathcal{N}\left(\boldsymbol{y}(t_*)|\{\tilde{\mu}_{j*}\}_{j=1}^{M}, \tilde{\sigma}_*^2\right) \right\rangle_{\mathcal{N}\left(\boldsymbol{y}(t_*)|\{\hat{\mu}_{j*}\}_{j=1}^{M}, \hat{\sigma}_*^2\right)} \end{aligned} \tag{42}$$

where T is the duration of the test sequence, and the quantities $\{\hat{\mu}_{j*}\}_{j=1}^{M}, \hat{\sigma}_*^2, \{\tilde{\mu}_{j*}\}_{j=1}^{M}$, and $\tilde{\sigma}_*^2$ are computed with the reservoir state values $\boldsymbol{\psi}(t_*)$ being obtained by means of a teacher-driven simulation of the employed reservoir (common for all models) for the considered test sequence. Apart from the ESGP, we also evaluated simple Gaussian mixture HMMs (GHMMs) [32], as well as Student's-t HMMs (SHMMs) [33] on the same task. The number of states and mixture components of these models were optimized as described in [33]. The obtained results are provided in Table V. As we observe, not only does the ESGP completely outperform conventional HMM formulations, but it also manages to yield better performance compared to the recently proposed SHMM approach.

TABLE V
SEQUENTIAL DATA CLASSIFICATION: AVERAGE ERROR RATE OF THE EVALUATED MODELS (TEXT-DEPENDENT SPEAKER IDENTIFICATION TASK)

Model	GHMM (three states/three component distributions)	SHMM (three states/three component distributions)	ESGP
Classification error rate	**1.93%**	**1.56%**	**0.99%**

TABLE VI
HUMAN MOTION MODELING: MISSING FRAMES RMSE

Video ID	DynGP	Ridge-regression-based ESN	ESGP	SVESM
35-03	49.68	62.55	32.59	35.12
12-02	54.96	63.14	45.32	57.88
16-21	78.05	98.74	59.03	69.07
12-03	63.63	72.12	46.25	53.88
07-01	84.12	121.47	77.34	95.44
07-02	80.77	100.94	73.88	79.21
08-02	95.52	120.45	101.54	100.12
08-01	82.66	152.44	118.0	133.65
Average	73.67	98.98	69.24	78.05

F. Human Motion Modeling

In this last experiment, we train the evaluated methods using four walking sequences from the Carnegie Mellon University (CMU) MoCap dataset [34]. The considered training sequences, corresponding to four different subjects, are obtained from the CMU database files: 35_02, 10_04, 12_01, and 16_15[2]. All are downsampled by a factor of 4. In the sequel, we use these models to generate the human pose information in the walking sequence videos 35-03, 12-02, 16-21, 12-03, 07-01, 07-02, 08-01, and 08-02 of the same database. Specifically, the trained models are presented with the first 10 samples from each considered video, and are asked to estimate the human pose in the rest of the video frames. To effect this, the inputs presented to the evaluated algorithms are the positions of the tracked human joints, and their output is the predicted joint positions at a time point of interest. In Table VI, we provide the RMSEs obtained by each one of the considered methods. As we observe, the ESGP performs better than all the rest of the evaluated methods both in average and in the most of the individual experimental cases considered here. Bearing also in mind that running these experiments took around 200 s in the case of the ESGP model, and more than 2 h in the case of the DynGP, it becomes apparent that the ESGP is a favorable alternative over DynGP in this application.

VI. CONCLUSION

In this paper, we proposed a new methodology for Bayesian modeling of sequential data, namely, the ESGP. The ESGP model can be viewed as a form of GPs, employing a special type of kernels *(reservoir kernels),* that allow the model to capture the temporal dynamics in a modeled dataset. Conversely, the ESGP can be viewed as a Bayesian treatment of ESNs, the predictions generated by which reduce to those obtained by simple ridge-regression-based ESNs in case a linear reservoir kernel is employed. Compared to existing ESN treatments, the major advantages of the proposed approach can be summarized in the following points:

1) our method provides a measure of confidence (or, better, uncertainty) on the generated predictions, the model predictive density, at the points of prediction;
2) our method is endowed with a marginal likelihood function, which can be used to conduct model parameters estimation by means of type-II maximum likelihood, thus in a much more computationally efficient way compared to the cross-validation techniques that ridge-regression-based ESNs actually rely on;
3) as experimentally demonstrated, the nonparametric Bayesian nature of our method allows increased predictive performance compared to existing ESN-based approaches.

Additionally, with respect to existing GP-based methodologies for sequential data modeling, such as the DynGP of [19], we observe that our method is overwhelmingly more computationally efficient, especially in cases where large data corpora have to be processed, whereas it performs at least comparably to the DynGP.

Few open issues of the ESGP that we aim to address in the near future include speeding up the ESGP inference algorithm by application of sparse approximations of the model covariance functions (e.g., [35]), and examining the utility of different kinds of reservoir kernels. The MATLAB implementation of the ESGP method shall be made available through the website of the authors: http://www.iis.ee.ic.ac.uk/~sotirios.

[2]Following the experimental setup of [19], model training was conducted using frames 55 to 338 of video 35_02, frames 222 to 499 of video 10_04, frames 22 to 328 of video 12_01, and frames 62 to 342 of video 16_15.

ACKNOWLEDGMENT

The authors would like to thank Y. Wu, Ph.D. student at the Electrical and Electronic Engineering Department, Imperial College London, London, U.K., for providing assistance in experimenting with the iCub platform, and obtaining the related datasets for the learning-to-grasp task.

REFERENCES

[1] H. T. Siegelmann and E. D. Sontag, "Turing computability with neural nets," *Appl. Math. Lett.*, vol. 4, no. 6, pp. 77–80, 1991.
[2] G. Deco and B. Schurmann, "Neural learning of chaotic dynamics," *Neural Process. Lett.*, vol. 2, no. 2, pp. 23–26, 1995.
[3] P. Tino, C. Schittenkopf, and G. Dorffner, "Financial volatility trading using recurrent neural networks," *IEEE Trans. Neural Netw.*, vol. 12, no. 4, pp. 865–874, Jul. 2001.
[4] F. Kianifardand and W. Swallow, "A review of the development and application of recursive residuals in linear models," *J. Amer. Stat. Assoc.*, vol. 91, no. 443, pp. 391–400, Mar. 1996.
[5] S. Haykin and J. Principe, "Making sense of a complex world [chaotic events modeling]," *IEEE Signal Process. Mag.*, vol. 15, no. 3, pp. 66–81, May 1998.
[6] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks," German Nat. Research Center Information Technology, Bremen, Germany, Tech. Rep. 148, 2001.

[7] W. Maass, T. Natschlaeger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Comput.*, vol. 14, no. 11, pp. 2531–2560, Nov. 2002.
[8] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Netw.*, vol. 20, no. 3, pp. 391–403, Apr. 2007.
[9] M. Lukosevicius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Comput. Sci. Rev.*, vol. 3, no. 3, pp. 127–149, Aug. 2009.
[10] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.
[11] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.
[12] M. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, Jun. 2001.
[13] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[14] A. Rodan and P. Tino, "Minimum complexity echo state network," *IEEE Trans. Neural Netw.*, vol. 22, no. 1, pp. 131–144, Jan. 2011.
[15] X. Yili, B. Jelfs, M. Van Hulle, J. Principe, and D. Mandic, "An augmented echo state network for nonlinear adaptive filtering of complex noncircular signals," *IEEE Trans. Neural Netw.*, vol. 22, no. 1, pp. 74–83, Jan. 2011.
[16] H. Jaeger, M. Lukosevicius, D. Popovici, and U. Siewert, "Optimization and applications of echo state networks with leaky-integrator neurons," *Neural Netw.*, vol. 20, no. 3, pp. 335–352, Apr. 2007.
[17] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, Apr. 2004.
[18] M. Buehner and P. Young, "A tighter bound for the echo state property," *IEEE Trans. Neural Netw.*, vol. 17, no. 3, pp. 820–824, May 2006.
[19] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 283–298, Feb. 2008.
[20] Z. Shi and M. Han, "Support vector echo-state machine for chaotic time-series prediction," *IEEE Trans. Neural Netw.*, vol. 18, no. 2, pp. 359–372, Mar. 2007.
[21] E. A. Antonelo, B. Schrauwen, and J. V. Campenhout, "Generative modeling of autonomous robots and their environments using reservoir computing," *Neural Process Lett.*, vol. 26, no. 3, pp. 233–249, 2007.
[22] M. Seeger, "Bayesian Gaussian process models: PAC-Bayesian generalisation error bounds and sparse approximations," Ph.D. dissertation, School Inf., Univ. Edinburgh, Edinburgh, U.K., 2003.
[23] M. F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Netw.*, vol. 6, no. 4, pp. 525–533, 1993.
[24] F. Wyffels, B. Schrauwen, and D. Stroobandt, "Stable output feedback in reservoir computing using ridge regression," in *Proc. 18th Int. Conf. Artif. Neural Netw.*, LNCS 5163. 2008, pp. 808–817.
[25] N. Lawrence. *Gaussian Process Software* [Online]. Available: http://staffwww.dcs.shef.ac.uk/people/n.lawrence/software.html
[26] C.-C. Chang and C.-J. Lin. (2001). *LIBSVM: A Library for Support Vector Machines* [Online]. Available: http://www.csie.ntu.edu.tw/∼cjlin/libsvm
[27] M. Henon, "A 2-D mapping with a strange attractor," *Commun. Math. Phys.*, vol. 50, no. 1, pp. 69–77, 1976.
[28] Y. Wu and Y. Demiris, "Efficient template-based path imitation by invariant feature mapping," in *Proc. IEEE Int. Conf. Robot. Biomimet.*, Dec. 2009, pp. 913–918.
[29] *The RobotCub Consortium* [Online]. Available: http://www.robotcub.org/
[30] M. Kudo, J. Toyama, and M. Shimbo, "Multidimensional curve classification using passing-through regions," *Pattern Recognit. Lett.*, vol. 20, nos. 11–13, pp. 1103–1111, Nov. 1999.
[31] A. Asuncion and D. Newman. (2007). *UCI Machine Learning Repository* [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html
[32] G. McLachlan and D. Peel, *Finite Mixture Models* (Probability and Statistics). New York: Wiley, 2000.
[33] S. P. Chatzis, D. I. Kosmopoulos, and T. A. Varvarigou, "Robust sequential data modeling using an outlier tolerant hidden Markov model," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 9, pp. 1657–1669, Sep. 2009.
[34] *The CMU MoCap Database* [Online]. Available: http://mocap.cs.cmu.edu/
[35] N. D. Lawrence, J. C. Platt, and M. I. Jordan, "Extensions of the informative vector machine," in *Deterministic and Statistical Methods in Machine Learning*, J. Winkler, N. D. Lawrence, and M. Niranjan, Eds. Berlin, Germany: Springer-Verlag, 2005, pp. 56–87.

Sotirios P. Chatzis (M'07) received the M.Sc. degree (five-year Diploma) in electrical and computer engineering in 2005, and the Ph.D. degree in machine learning in 2008, both from the National Technical University of Athens, Athens, Greece, in 2005.

He is currently a Post-Doctoral Researcher in the Department of Electrical and Electronic Engineering, Imperial College London, London, U.K. His current research interests include machine learning theory and methodologies with special focus on hierarchical Bayesian models, reservoir computing, robot learning by demonstration, copulas, quantum statistics, Bayesian nonparametrics, and artificial creativity. In the last five years, he has first-authored 28 papers published in the most prestigious journals of his research field.

Dr. Chatzis was supported by the Bodossaki Foundation, Athens, and the Greek Ministry for Economic Development during the Ph.D. program, and he was awarded the Dean's Scholarship for Ph.D. studies, being the best performing Ph.D. student of his class.

Yiannis Demiris (SM'07) received the B.Sc. and Ph.D. degrees in intelligent robotics from the University of Edinburgh, Edinburgh, U.K.

He is a Senior Lecturer with the Department of Electrical and Electronic Engineering, Imperial College London, London, U.K., where he directs the Bioinspired Assistive Robots and Teams (BioART) Laboratory doing research in cognitive systems, assistive robotics, multi-robot systems, and robot human interaction.

Dr. Demiris has organized several international workshops on Robot Learning, Bioinspired Machine Learning, and Epigenetic Robotics. He has received fellowships from the AIST-MITI in Japan and the European Science Foundation. He was the Chair of the IEEE International Conference on Development and Learning 2007 and the Program Chair of the ACM/IEEE International Conference on Human-Robot Interaction in 2008.