Iterative Temporal Learning and Prediction with the Sparse Online Echo State Gaussian Process

Harold Soh
Imperial College London
Email: haroldsoh@imperial.ac.uk

Yiannis Demiris
Imperial College London
Email: y.demiris@imperial.ac.uk

***Abstract*—In this work, we contribute the *online echo state gaussian process* (OESGP), a novel Bayesian-based online method that is capable of iteratively learning complex temporal dynamics and producing predictive distributions (instead of point predictions). Our method can be seen as a combination of the echo state network with a sparse approximation of Gaussian processes (GPs). Extensive experiments on the one-step prediction task on well-known benchmark problems show that OESGP produced statistically superior results to current online ESNs and state-of-the-art regression methods. In addition, we characterise the benefits (and drawbacks) associated with the considered online methods, specifically with regards to the trade-off between computational cost and accuracy. For a high-dimensional action recognition task, we demonstrate that OESGP produces high accuracies comparable to a recently published graphical model, while being fast enough for real-time interactive scenarios.**

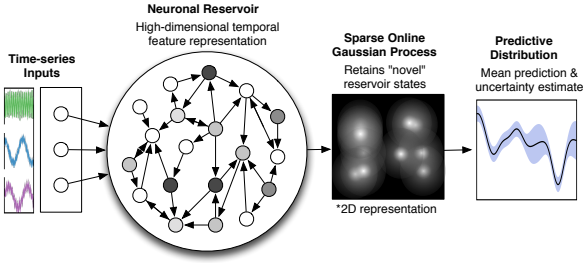


Fig. 1. The Online Echo State Gaussian Process which learns online from temporal sequences and produces predictive distributions.

I. Introduction

The importance of learning and prediction in temporal domains is exhibited in a wide variety of scientific and engineering fields. For example, the characterisation of dynamical systems helps us understand (and predict) phenomena ranging from human motion to the transmission of information packets across the world-wide-web. As such, a multitude of temporal modelling approaches have been developed over the decades, such as autoregressive techniques and more complex state-space methods. In the computational intelligence community, recurrent neural networks (RNNs) represent the standard approach. Unfortunately, early RNNs were plagued by training difficulties which presented scaling issues and limited their wide-spread use [1].

In recent years, advances in *reservoir computing* [2] have rekindled research into RNNs. In particular, echo state networks (ESNs) [3] have demonstrated not only significant performance gains, but also simplified training over traditional RNN models; compared to canonical training where all weights in a network are adapted, only the mapping from a fixed neural network (the reservoir) are trained. The structure of reservoirs has a substantial impact on performance, resulting in an abundance of fresh research into optimising reservoir topology (e.g, [4], [5]).

Less attention, however, has been paid to echo-state networks that can learn iteratively as data becomes available. This is unfortunate because online algorithms are important for both theoretical and applied work; for example, in humanoid robotics, online learning methods provide adaptive controllers that adjust to changing physical or human factors [6]. A second related, but distinct, issue is that applications often call for models to give confidences (or indications of uncertainty) along with predictions. Uncertainty estimates are relevant in real-time decision making to compute probable outcomes and also active-learning [7] where data is scarce or expensive to obtain.

In this work, we address both issues simultaneously by contributing the *online echo state gaussian process* (OESGP); a non-parametric iterative temporal-learning method (illustrated in Fig. 1). To reduce computational and storage costs, OESGP stores only "informative" or "novel" neural states; non-novel states are absorbed but do not increase model size. We achieve this using online sparse approximations [8] for gaussian processes, enabling fast iterative learning and prediction given large amounts of sequential data.

Unlike current online ESNs trained using recursive least squares (RLS) [9] and stochastic gradient descent (SGD) [10], our Bayesian-based formulation gives predictive distributions instead of point-predictions. One can view the OESGP as an online variant of the recently-proposed echo state gaussian process (ESGP) [11], a combination of the ESN and gaussian process (GP) approaches. The use of kernels allows for non-linear mappings between reservoir states and desired outputs, permitting greater flexibility in modelling and adapting to dynamical systems. The ESGP, however, has high $O(n^3)$ complexity, making it too expensive for many real-time applications. As we will show, OESGP's computational cost can be controlled by limiting the number of stored neural states. Comparing the OESGP to many existing online regres-

sion algorithms (e.g., Locally Weighted Projection Regression (LWPR) [6] and Kernel RLS [12]), our ESN-based method takes into account temporal dependencies without explicitly specified embeddings.

The remaining sections of this paper are organised as follows. The next section presents related background work; in particular, online ESNs and the ESGP. In Section III, we describe our main contribution, the OESGP. Section IV describes our experimental results on our benchmark datasets and Section V describes our action recognition classifier. Section VI discusses the trade-off between computational cost and accuracy as well as some avenues for future work. Finally, we conclude this paper in Section VII with a summary as well as some final remarks.

II. Background

This section provides the necessary background for our work; we first give an overview of ESNs, with a focus on online variants proposed by Jaeger [9] and Kountouriotis *et al*. [10], followed by a brief review of the ESGP [11].

A. Online Echo State Networks

Before the advent of ESNs, recurrent neural networks were typically trained by adapting *all weights* through gradient descent methods such as backpropagation through time (BPTT) [13], [14] and real-time recurrent learning (RTRL) [15]. Unfortunately, these methods were slow to converge (due to vanishing gradients), relatively difficult to implement and subject to bifurcations.

The echo state network proposed by Jaeger [3] is a novel approach to RNN training and architecture that addresses all three issues. Instead of adapting all network weights, *only the output weights are trained*. The basic notion is to drive a randomly-generated fixed RNN (called the *reservoir*) using the input signal and then derive the output via some combination of the reservoir units (e.g., using standard linear regression). More precisely, the state of the reservoir is updated during training:

$$\mathbf{x}_{t+1} = (1-\gamma)h(\mathbf{W}\mathbf{x}_t + \mathbf{W}_i\mathbf{u}_{t+1} + \mathbf{W}_b\mathbf{d}_t) + \gamma\mathbf{x}_t \quad (1)$$

where $\mathbf{x}_t$ is the state of the reservoir units at time t, $\mathbf{u}_t$ is the input, $h(\cdot)$ is the activation function, $\mathbf{d}_t$ is the desired output, $\mathbf{W}$ is reservoir weight matrix, $\mathbf{W}_i$ is the input weight matrix, $\mathbf{W}_b$ is the output feedback weight matrix, and γ is the leak (or retainment) rate. After training, the update equation becomes:

$$\mathbf{x}_{t+1} = (1-\gamma)h(\mathbf{W}\mathbf{x}_t + \mathbf{W}_i\mathbf{u}_{t+1} + \mathbf{W}_b\mathbf{y}_t) + \gamma\mathbf{x}_t \quad (2)$$

and the predicted outputs $\mathbf{y}_t$ are obtained using:

$$\mathbf{y}_{t+1} = \mathbf{W}_o\boldsymbol{\psi}_{y+1} \quad (3)$$

where $\mathbf{W}_o$ is the linear output weight matrix and

$$\boldsymbol{\psi}_{t+1} \triangleq [\mathbf{x}_{t+1}; \mathbf{u}_{t+1}] \quad (4)$$

is the augmented reservoir state and input vector.

In the case of offline training, the augmented reservoir states, $\boldsymbol{\psi}$, are gathered and regressed against the desired outputs. For *online* training, this regression can be performed as the reservoir evolves over time via RLS [9]. Briefly, RLS is an adaptive filter which finds the output weights that minimise the least squares error function. In its basic form, RLS (applied to the ESN) consists of the following iterative updates:

$$\begin{aligned}
\varrho_{t+1} &= d_{t+1} - \boldsymbol{\psi}_{t+1}^{\mathrm{T}}\mathbf{w}_{o,t} \\
\mathbf{g}_{t+1} &= \mathbf{P}_t\boldsymbol{\psi}_{t+1}(\lambda + \boldsymbol{\psi}_{t+1}^{\mathrm{T}}\mathbf{P}_t\boldsymbol{\psi}_{t+1})^{-1} \\
\mathbf{P}_{t+1} &= \lambda^{-1}\mathbf{P}_t - \mathbf{g}_{t+1}\boldsymbol{\psi}_{t+1}^{\mathrm{T}}\lambda^{-1}\mathbf{P}_t \\
\mathbf{w}_{o,t+1} &= \mathbf{w}_{o,t} + \varrho_t\mathbf{g}_{t+1}
\end{aligned} \quad (5)$$

where $\mathbf{w}_{o,t}$ is a row of the output weights matrix at time t and λ is the forgetting factor. At $t=0$, $\mathbf{w}_{o,0}=0$ and $\mathbf{P}_0 = \delta^{-1}\mathbf{I}$ where δ is user-defined. From the equations, readers may recognise RLS as a special-case of the popular Kalman filter. Using the RLS-ESN, Jaeger demonstrated online adaptation of the ESN for a system identification task (a tenth-order NARMA problem). In general, the RLS-ESN exhibits fast convergence but is subject to numerical instability[1] and is computationally expensive compared to other adaptive filters; each RLS update is on the order of $O(N_{\boldsymbol{\psi}}^2)$ where $N_{\boldsymbol{\psi}}$ is length of $\boldsymbol{\psi}$.

As a cheaper $O(N_{\boldsymbol{\psi}})$ alternative, Kountoriotis *et al.* [10] proposed updating the output weights directly via stochastic gradient descent (SGD) or equivalently, Least Means Squares (LMS), i.e.:

$$\mathbf{W}_{o,t+1} = \mathbf{W}_{o,t} + \eta(\mathbf{d}_t - \mathbf{y}_t)\mathbf{x}_t \quad (6)$$

where η is the learning rate. The authors illustrated the SGD-ESN was sufficient to perform multi-step tracking of a three-dimensional Lorenz system. That said, convergence performance of the SGD approach is not only heavily dependent on η but is also negatively impacted by the eigenvalue spread of the reservoir cross-correlation matrix [2].

The standard ESNs covered thus far produce point predictions but it is often desirable to obtain the uncertainty associated with the predicted outputs. In the next section, we give an overview of the ESGP, a recently proposed (offline) echo-state method that achieves this by delivering predictive distributions.

B. The Echo State Gaussian Process

The ESGP proposed by Chatzis and Demiris [11] is a Bayesian formulation of the standard echo-state network, based on Gaussian processes (GP). Given an observation space with elements $\mathbf{x} \in \mathcal{X}$, a GP is a set of random variables whereby any finite subset has a joint Gaussian distribution [16]. It is completely specified by its mean function,

$$m(\mathbf{x}) = \mathbb{E}[f(x)] \quad (7)$$

and its covariance function,

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \quad (8)$$

[1]As discussed in [9], numerical instability can be partially resolved using noise insertion.

and since it is often assumed that $m(\mathbf{x}) = 0$, the GP is written as:

$$f(\mathbf{x}) \sim \mathcal{N}(0, k(\mathbf{x}, \mathbf{x}')). \tag{9}$$

Given a set of identically distributed samples $(\mathbf{x}_i, \tilde{y}_i) \in \mathcal{D}$ where $\tilde{y}_i$ are the observed target values, let $\mathbf{K}$ be the matrix of covariances between the $N = |\mathcal{D}|$ training points, i.e. $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)]$. Also, let $\mathbf{k}(x') \triangleq [k(\mathbf{x}_i, \mathbf{x}')]_{i=1}^N$. If we assume that the observed signals were corrupted by additive independent white noise, $\epsilon \sim \mathcal{N}(0, \sigma^2)$, the GP predictive distribution for a new input point $\mathbf{x}_*$ is given by:

$$p(f_*|\mathbf{x}_*, \mathcal{D}) = \mathcal{N}(f_*|\mu_*, \sigma_*^2) \tag{10}$$

where

$$\mu_* = \mathbf{k}(\mathbf{x}_*)^{\mathrm{T}}(\mathbf{K} + \sigma^2\mathbf{I}_N)^{-1}\mathbf{y} \tag{11}$$

and

$$\sigma_*^2 = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}(\mathbf{x}_*)^{\mathrm{T}}(\mathbf{K} + \sigma^2\mathbf{I}_N)^{-1}\mathbf{k}(\mathbf{x}_*). \tag{12}$$

As can be seen from the above discussion, GPs provide probabilistic outputs (in the form of normal distributions) instead of simple point predictions. Returning to ESNs, recall from (3) that each echo state output is produced by multiplying a row from the readout matrix $\mathbf{W}_o$ and $\boldsymbol{\psi}$ (the augmented network-state and input vector). If we consider the imposition of a Gaussian prior over each row of weights, $\mathbf{w}_j \sim \mathcal{N}(0, \mathbf{I})$, we can derive that the distributions of the ESN outputs yield a GP for each output:

$$[y_{j,t}]_{t=t_1}^{t_T} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_r) \tag{13}$$

where $\mathbf{K}_r = [k_r(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)]_{i,j=t_1}^{t_T}$ and $k_r(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)$ is the chosen *reservoir kernel function*. For example, the popular radial basis function (RBF) kernel is given by: $k_r(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j) \triangleq \exp\left(b||\boldsymbol{\psi}_i - \boldsymbol{\psi}_j||^2\right)$. Based on the above formulation, we observe that the ESGP uses a covariance function which is a kernel function on the ESN state, thus capturing temporal relationships between sequential observations [11]. Theoretical subtleties aside, one can consider the ESGP as an echo state network where training is performed with a GP instead of linear regression.

The ESGP presented is a generalisation of ESNs trained via linear/ridge regression; the mean-prediction of the ESGP with a linear kernel is identical to the one produced by a ESN trained using ridge-regression. In [11], the authors found the ESGP to be highly effective on a variety of benchmark and real-world tasks, achieving high accuracies with a reasonable increase in training cost.

Turning our attention to online learning, one can consider a direct application of the ESGP by "growing" the kernel matrix as new samples are processed. However, two major issues make this approach infeasible in practice. First, the ESGP incurs a high computational cost; the matrix inversion needed in the predictions are $O(n^3)$ for a $n \times n$ kernel matrix. This is likely too slow for real-time applications. Second, $\mathbf{K}_r$, will grow quadratically and *unbounded*, thereby straining both storage and computational resources.

III. The Online Echo State Gaussian Process

In this section, we present our proposed *online echo state gaussian process* (OESGP), a sparse variant of the full ESGP designed to solve the aforementioned difficulties of high computation and storage costs. Our approach is based on the sparse online GP (SOGP) proposed by Csató and Opper [17], [8] and we closely follow their treatment. While other sparse representations of GPs exist (see [18], [16] for an overview), they generally require that the entire dataset is available during training, making them inappropriate for online learning.

As a guide, we first determine how to perform fast successive updates to the ESGP as new data points arrive. Then, we solve the unbounded storage problem by keeping only "novel" reservoir states up to some maximum capacity.

A. Bayesian Online Learning and the Projected Process Approximation

Taking the first problem, let us denote $\mathbf{u}_{t+1}$ as the input into the reservoir, $\tilde{y}_{t+1}$ as the observed output signal, d_{t+1} as the true noise-free output, $\mathbf{x}_{t+1}$ as the updated reservoir state and $\boldsymbol{\psi}_{t+1} \triangleq [\mathbf{x}_{t+1}; \mathbf{u}_{t+1}]$. Our goal is to incrementally update the ESGP given $(\mathbf{u}_{t+1}, \tilde{y}_{t+1})$. Applying Bayesian online learning for regular GPs [19], [8] to our specific case, our approach consists of two basic steps:

1) **Update the ESN state** using (1) to derive the new composite state $\boldsymbol{\psi}_{t+1}$.
2) **Update the model posterior** given $(\boldsymbol{\psi}_{t+1}, \tilde{y}_{t+1})$ and **project the posterior** onto the closest GP.

While step 1 is easily achieved, step 2 deserves more discussion. Let us assume we have an ESGP at time t. We incorporate a new datapoint into the ESGP by performing a Bayesian update to yield a posterior:

$$\hat{p}(\mathbf{f}|\tilde{y}_{t+1}) = \frac{P(\tilde{y}_{t+1}|f(\boldsymbol{\psi}_{t+1}))p_t(\mathbf{f})}{\langle P(\tilde{y}_{t+1}|f(\boldsymbol{\psi}_{t+1}))p_t(\mathbf{f})\rangle_t}. \tag{14}$$

Unfortunately, the update cannot be applied repeatedly since it yields a posterior process that is typically non-Gaussian with intractable integrals. The way we get around this is to *project* the process onto the closest GP where "closest" is measured via the Kullback-Leibler divergence, $KL(\hat{p}_t||q)$, and q is our desired approximation. Minimising the KL divergence is equivalent to matching the first two moments of $\hat{p}$ and q, which can be performed analytically. The update equations in their "natural parameterisation" forms are given by[2]:

$$m_t(\boldsymbol{\psi}) = \boldsymbol{\alpha}_t^{\mathrm{T}}\mathbf{k}_r(\boldsymbol{\psi}) \tag{15}$$

$$k_t(\boldsymbol{\psi}, \boldsymbol{\psi}') = k_r(\boldsymbol{\psi}, \boldsymbol{\psi}') + \mathbf{k}_r(\boldsymbol{\psi})^{\mathrm{T}}\mathbf{C}_t\mathbf{k}_r(\boldsymbol{\psi}') \tag{16}$$

where $\boldsymbol{\alpha}$ vector and $\mathbf{C}$ are updated using:

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t + b_1(\mathbf{C}_t\mathbf{k}_{r,t+1} + \mathbf{e}_{t+1}) \tag{17}$$

$$\mathbf{C}_{t+1} = \mathbf{C}_t + b_2(\mathbf{C}_t\mathbf{k}_{r,t+1} + \mathbf{e}_{t+1})(\mathbf{C}_t\mathbf{k}_{r,t+1} + \mathbf{e}_{t+1})^{\mathrm{T}} \tag{18}$$

[2]The proofs for this derivation can be found in [8].

where $\mathbf{k}_{r,t+1} = [k_r(\boldsymbol{\psi}_1, \boldsymbol{\psi}_{t+1}), ..., k_r(\boldsymbol{\psi}_t, \boldsymbol{\psi}_{t+1})]$, $\mathbf{e}_{t+1}$ is the $t+1^{\text{th}}$ unit vector and the scalar coefficients b_1 and b_2 are given by:

$$b_1 = \partial_{f_t} ln \langle P(\tilde{y}_{t+1}|f(\boldsymbol{\psi}_{t+1}))\rangle_t \tag{19}$$

$$b_2 = \partial^2_{f_t} ln \langle P(\tilde{y}_{t+1}|f(\boldsymbol{\psi}_{t+1}))\rangle_t \tag{20}$$

In the case of regression with gaussian noise, note that b_2 does not depend on the outputs $\tilde{y}_{t+1}$ so, only a single matrix $\mathbf{C}$ needs to be updated regardless of the output dimensionality. Although the "full update" equations (15)-(16) allow us to update the ESGP sequentially, $\boldsymbol{\alpha}$ and $\mathbf{C}$ increase with the number of processed samples.

B. Maintaining Sparsity

The core concept here is to limit the number of the reservoir states retained (called the *basis vectors* (BV), $\mathbf{b} \in \mathcal{B}$) using a scoring function that computes the "novelty" of the state $\boldsymbol{\psi}_{t+1}$. Two basic steps are involved in maintaining the sparsity:

1) **Compute the score** of $\boldsymbol{\psi}_{t+1}$ and if the score is higher than some threshold, perform an update using (15)-(16).
2) **Maintain the size of** $\mathcal{B}$ by removing the lowest scoring BV if $|\mathcal{B}|$ exceeds some predefined capacity.

Different scoring functions are possible, e.g., Seeger [20] uses a low-cost approximation of the information gain while Keerthi [21] uses a more expensive, but more accurate, matching pursuit approach. In this work, we follow [17], [8] and use the function:

$$\gamma(\boldsymbol{\psi}_{t+1}) = k_r(\boldsymbol{\psi}_{t+1}, \boldsymbol{\psi}_{t+1}) - \mathbf{k}^{\mathrm{T}}_{\mathcal{B},t+1}\mathbf{K}^{-1}_{\mathcal{B},t}\mathbf{k}_{\mathcal{B},t+1} \tag{21}$$

where $\mathbf{k}_{\mathcal{B},t+1} = [k_r(\mathbf{b}_i, \boldsymbol{\psi}_{t+1})]_{\mathbf{b}_i \in \mathcal{B}}$ and $\mathbf{K}^{-1}_{\mathcal{B},t} = [k_r(\mathbf{b}_i, \mathbf{b}_j)]_{\mathbf{b}_i, \mathbf{b}_j \in \mathcal{B}}$. If $\gamma(\boldsymbol{\psi}_{t+1})$ is below some constant threshold, ϵ_γ (10^{-6} in our work), then we do *not* perform the full update. Instead, we perform an *approximate update* using (17) and (18) with the only change being that we use:

$$\hat{\mathbf{e}}_{t+1} = \mathbf{K}^{-1}_{\mathcal{B},t}\mathbf{k}_{r,t+1} \tag{22}$$

instead of the unit vector $\mathbf{e}_{t+1}$. This update does not increase the size $\mathcal{B}$ but does absorb states which are not included. This operation may appear expensive since it involves computing the inverse of $\mathbf{K}_{\mathcal{B}}$. However, this inversion can be performed iteratively, i.e., $\mathbf{K}^{-1}_{\mathcal{B},t+1} = \mathbf{K}^{-1}_{\mathcal{B},t} + \gamma^{-1}_{t+1}(\hat{\mathbf{e}}_{t+1} - \mathbf{e}_{t+1})(\hat{\mathbf{e}}_{t+1} - \mathbf{e}_{t+1})^{\mathrm{T}}$ and hence, is linear.

Since we also limit the maximum size (*capacity*) of $\mathcal{B}$, it may be necessary to delete a basis vector. Assume that we have just added a new BV. We then score each $\mathbf{b}_i \in \mathcal{B}$ using the scoring function:

$$\epsilon_i = \frac{|\boldsymbol{\alpha}_{t+1}(i)|}{\mathbf{K}^{-1}_{\mathcal{B},t+1}(i,i)} \tag{23}$$

and remove the lowest scoring BV using a reduced update of our model. Suppose we wish to remove the j^{th} BV. Let us define $\boldsymbol{\alpha}'$ as the vector $\boldsymbol{\alpha}_{t+1}$ with the element $\alpha^* = \boldsymbol{\alpha}_{t+1}(j)$ removed. We define $\mathbf{C}'$ as the matrix $\mathbf{C}_{t+1}$ without the j^{th} row and column and $c^* = \mathbf{C}_{t+1}(j,j)$. The column vector $\mathbf{C}^*$ is the j^{th} row without c^*. Let $\mathbf{Q} = \mathbf{K}^{-1}_{\mathcal{B},t+1}$ and $\mathbf{Q}'$, $\mathbf{Q}^*$, q^* be similarly defined as for $\mathbf{C}$. Then, our reduced update equations are given by:

$$\hat{\boldsymbol{\alpha}}_{t+1} = \boldsymbol{\alpha}' - \alpha^* \frac{\mathbf{Q}^*}{q^*} \tag{24}$$

$$\hat{\mathbf{C}}_{t+1} = \mathbf{C}' + c^* \frac{\mathbf{Q}^*\mathbf{Q}^{*\mathrm{T}}}{q^{*2}} - \frac{1}{q^*}\left(\mathbf{Q}^*\mathbf{C}^{*\mathrm{T}} + \mathbf{C}^*\mathbf{Q}^{*\mathrm{T}}\right) \tag{25}$$

$$\hat{\mathbf{Q}}_{t+1} = \mathbf{Q}' - \frac{\mathbf{Q}^*\mathbf{Q}^{*\mathrm{T}}}{q^*} \tag{26}$$

This completes our discussion of the main aspects of the sparse approximation.

C. Training Summary and Making Predictions

To summarise, training the OESGP consists of four basic steps:

1) **Update the ESN state** using 1 to derive the new composite state $\boldsymbol{\psi}_{t+1}$.
2) **Compute the score** of $\boldsymbol{\psi}_{t+1}$ using (21).
3) **Perform a full update** using (15)-(16) if the score is higher than a pre-defined threshold ϵ_γ. Otherwise, **perform an approximate update** by substituting $\mathbf{e}_{t+1}$ with $\hat{\mathbf{e}}_{t+1}$ (22).
4) **Maintain the size of** $\mathcal{B}$ by removing the lowest scoring BV using (24)-(26) if $|\mathcal{B}|$ exceeds some predefined capacity.

Making predictions with the OESGP is straightforward with the mean of the predictive distribution given by:

$$\mu_* = \mathbf{k}_{\mathcal{B},t}(\boldsymbol{\psi}_{t*})^{\mathrm{T}}\boldsymbol{\alpha}_t \tag{27}$$

and variance:

$$\sigma^2_* = k_r(\boldsymbol{\psi}_{t*}, \boldsymbol{\psi}_{t*}) + \mathbf{k}_{\mathcal{B},t}(\boldsymbol{\psi}_{t*})^{\mathrm{T}}\mathbf{C_t}\mathbf{k}_{\mathcal{B},t}(\boldsymbol{\psi}_{t*}) \tag{28}$$

Conceptually, our method can be seen as an online, iterative version of the ESGP using the sparse approximations developed by Csató and Opper [8], [17]. Compared to ESGP, our variant has a lower computational complexity of $O(s^2_{\mathcal{B}} + N_{\boldsymbol{\psi}} s_{\mathcal{B}})$ time where $s_{\mathcal{B}}$ is the maximum BV set size, typically chosen based on available computational resources.

IV. Empirical Results

In this section, we report on experiments designed to investigate the performance of OESGP, relative to the other online ESNs (SGD-ESN and RLS-ESN) and state-of-the-art online regression algorithms (e.g., the widely-used LWPR [6]).

A. Implementation and Setup

Our OESGP was coded in C++ using the Eigen [22] and SOGP libraries [23]. Our source codes are available online [24]. The LWPR results were obtained using the *lwpr* library with MATLAB bindings [25]. Runs were conducted on a 2.8Ghz Intel Core i7 processor, with each run repeated 30 times. All online ESNs were initialised so that the same reservoir was used for a given sequence.

For RLS and SGD parameters, we used a grid search on a subset of the data (5000 samples) and selected the parameters with the lowest RMSE (on the last 20% of the sequence

TABLE I
ESN AND ALGORITHM PARAMETERS FOR THE PROBLEMS CONSIDERED IN THIS STUDY.

Problem	Reservoir size	Spectral radius	Connectivity	SGD-ESN η	RLS-ESN (δ, λ)	OESGP (b, σ)
Mackey-Glass	100	0.99	0.1	0.05	$(10^{-8}, 0.99999)$	(0.05, 0.005)
Henon	100	0.90	0.1	0.01	$(10^{-9}, 0.999)$	(0.1, 0.1)
Laser	100	0.90	0.1	0.05	$(10^{-9}, 0.999999)$	(0.05, 0.002)
Ikeda	200	0.90	0.1	0.001	$(10^{-8}, 0.999999)$	(0.1, 0.001)
Lorenz	100	0.99	0.2	0.05	$(10^{-1}, 0.999999)$	(0.1, 0.05)
NP1	50	0.90	0.1	0.01	$(10^{-3}, 0.999)$	(0.25, 0.0001)
NP2	100	0.90	0.1	0.01	$(10^{-3}, 0.99999)$	(0.5, 0.1)

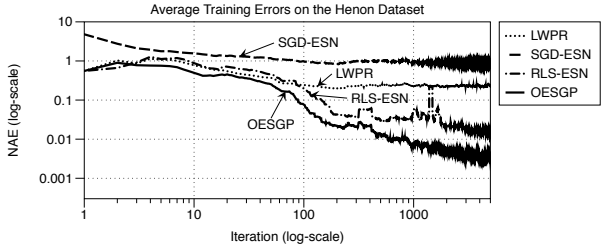


Fig. 2. NAE convergence profiles during training on the Henon dataset (averaged across 30 runs and smoothed using a 100-point moving average).

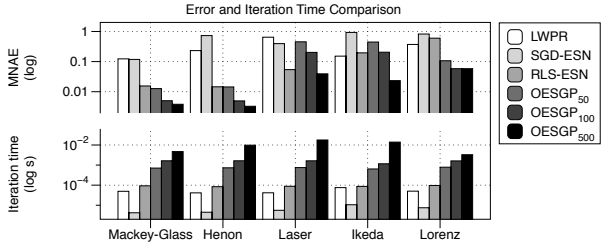


Fig. 3. MNAE and iteration time on the one-step prediction task on the "clean" benchmark problems.

averaged over 10 independent runs). LWPR parameters were set to recommended defaults with adaptation enabled [26]. For OESGP, we used the RBF kernel and varied capacities, $s_{\mathcal{B}} = 50, 100$ and 500. Complete parameter listings for the underlying ESNs and algorithms for each problem are given in Table I.

B. One-Step Prediction on Benchmark Problems

We begin by presenting empirical results for the classic one-step prediction task on well-known benchmark problems, i.e., the Mackey-Glass, Henon, Lorenz and Ikeda dynamical systems, along with the Sante-Fe Institute (SFI) laser competition dataset [27]. Note that the Mackey-Glass, Ikeda, and Lorenz datasets were treated with the tangent-hyperbolic transform, as in [11]. In addition to these "clean" sequences, we include two systems with noisy observations, NP1 and NP2 [28]. For NP1, the inputs consisted of the pair (y_{t-1}, y_t) (consistent with [28] though not strictly necessary for the ESNs). We used standard equations for the dynamical and our scripts for generating the sequences are available [24].

Each problem consisted of sequences of 10^5 samples where training was performed for only half the sequence (predictions were still carried out during training), after which the algorithms were expected to produce predictions given inputs but without any future training. Methods were compared using two error functions computed on the final 50% of the sequence across all the runs, i.e., the mean normalised absolute error, $\text{MNAE} = \mathbb{E}[(sT_e)^{-1} \sum_{t=t_s}^{T_s} \sqrt{(d_t - \hat{d}_t)^2}]$, and the root-mean-square prediction error, $\text{RMSE} = \mathbb{E}[T_e^{-1} \sum_{t=t_s}^{T_s} \sqrt{(d_t - \hat{d}_t)^2}]$, where $\hat{d}_t$ is the predicted output at time t, $t_s = 5001$, $T_s = 10000$, s^2 is the empirical variance of the desired target signal and T_e is the length of the testing sequence. For multi-dimensional outputs, we averaged the scores over all outputs. Statistical significance between the error distributions generated by the online ESNs were tested using the Wilcoxon signed-rank test.

A summary of our results are shown in Table II. In general, we observed that OESGP (with 100 or 500 BVs) performed remarkably well, achieving the lowest MNAE scores for *all* the benchmark problems (results statistically significant at $p < 0.0001$). Compared to RLS-ESN, OESGP_{500} attains MNAE and RMSE scores that are up to 90% lower. In addition, the training error profiles showed that the convergence rate for OESGP was fast relative to the other algorithms; for example, on the Henon problem (see Fig. 2), a log-log fit gave a steeper slope (rate) of -0.53 for OESGP and -0.39 for RLS-ESN. Among the online ESNs, SGD-ESN obtained the worst error scores, evidently hampered by its inability to converge to proper weights. LWPR did not fare much better for many of the problems since it was not able to take into account temporal relationships.

Figure 3 summarises the mean iteration times where we observed a consistent pattern where SGD-ESN was the fastest method followed by LWPR, then RLS-ESN and finally OESGP. This was not surprising because training and predictions (at the pre-set reservoir size of 100 for most problems) are more expensive for OESGP compared to $O(N_{\boldsymbol{\psi}}^2)$ for RLS-ESN and $O(N_{\boldsymbol{\psi}})$ for SGD-ESN. LWPR, being a local method, also has favourable computational costs. That said, even when using only 50 BVs, OESGP achieves better scores compared to the other algorithms for the Mackey-Glass, Henon, Lorenz and NP2 problems. At this level of sparsity, each iteration remains reasonably fast at $\approx 10^{-3}$ while achieving better accuracy and providing uncertainty values.

Focussing on the noisy problems NP1 and NP2, we observed that OESGP performs marginally better than RLS-ESN on the NP1 problem and significantly better than all other

TABLE II
MNAE (TOP) AND RMSE (BOTTOM) SCORES FOR THE ONE-STEP PREDICTION TASK ON BENCHMARK AND NOISY PROBLEMS. STANDARD DEVIATIONS ARE SHOWN IN BRACKETS AND LOWEST ERROR SCORES ARE IN **BOLD**.

Problem	LWPR	SGD-ESN	RLS-ESN	OESGP$_{50}$	OESGP$_{100}$	OESGP$_{500}$
Mackey-	0.1232 (0.0008)	0.1181 (0.0718)	0.0155 (0.0471)	0.0127 (0.0021)	0.0050 (0.0004)	**0.0038** (0.0005)
Glass	0.0314 (0.0000)	0.0350 (0.0222)	0.0043 (0.0130)	0.0035 (0.0007)	0.0014 (0.0001)	**0.0011** (0.0001)
Henon	0.2317	0.7306 (0.0380)	0.0145 (0.0035)	0.0144 (0.0014)	0.0049 (0.0004)	**0.0033** (0.0001)
	0.2046	0.6064 (0.0306)	0.0135 (0.0032)	0.0143 (0.0013)	0.0055 (0.0003)	**0.0039** (0.0001)
Laser	0.6456	0.3953 (0.0307)	0.0541 (0.0020)	0.4538 (0.0761)	0.2024 (0.0455)	**0.0392** (0.0045)
	0.1458	0.1075 (0.0062)	**0.0198** (0.0011)	0.1208 (0.0254)	0.0569 (0.0117)	0.0210 (0.0017)
Ikeda	0.1517	0.9311 (0.0358)	0.1951 (0.0083)	0.4463 (0.0598)	0.2051 (0.0508)	**0.0233** (0.0040)
	0.0832	0.5039 (0.0218)	0.1177 (0.0048)	0.3049 (0.0631)	0.1418 (0.0501)	**0.0253** (0.0047)
Lorenz	0.3712	0.8230 (0.0301)	0.5997 (0.0820)	0.1066 (0.0414)	**0.0581** (0.0062)	0.0582 (0.0067)
	0.0494	0.1118 (0.0040)	0.0808 (0.0120)	0.0182 (0.0049)	**0.0133** (0.0012)	0.0133 (0.0013)
NP1	0.0859 (0.0016)	0.2888 (0.0658)	0.0580 (0.0023)	0.1308 (0.0189)	0.0596 (0.0018)	**0.0528** (0.0009)
	0.0910 (0.0013)	0.2502 (0.0497)	0.0572 (0.0019)	0.1451 (0.0264)	0.0589 (0.0016)	**0.0533** (0.0013)
NP2	0.4742 (0.0110)	0.9586 (0.3606)	0.8940 (0.4376)	0.1348 (0.0118)	**0.1325** (0.0139)	0.1354 (0.0117)
	0.2598 (0.0027)	0.4580 (0.1437)	0.3950 (0.1667)	0.0804 (0.0052)	**0.0797** (0.0056)	0.0807 (0.0053)

TABLE III
NMSE SCORES FOR THE ONE-STEP PREDICTION TASK ON NOISY PROBLEMS NP1 AND NP2. LOWEST ERROR SCORES ARE IN **BOLD**.

	NORMA	KNLMS	SSP	KRLS	OESGP$_{50}$	OESGP$_{100}$	OESGP$_{500}$
NP1	0.1051	0.0197	0.0184	0.0173	0.0408 (0.0150)	0.0065 (0.0004)	**0.0053** (0.0003)
NP2	0.56	0.20	0.21	0.17	0.0233 (0.0112)	**0.0229** (0.0110)	0.0236 (0.0114)

algorithms on the NP2 problem. To compare OESGP against recently published results demonstrating the performance of state-of-the-art online methods (KNLMS [28], KRLS [12], NORMA [29] and sparse sequential projection (SSP) [30]), we computed the normalised mean-square prediction error, NMSE $= \mathbb{E}[\sum_{t=t_s}^{T_s}(d_t - \hat{d}_t)^2 / \sum_{t=t_s}^{T_s} d_t^2]$ over testing portion (from sample 5001 onwards). Our error scores along with results reproduced from [28] are shown in Table III. We observed that OESGP's scores are an improvement of almost an order of magnitude over the published results.

V. CASE STUDY: ONLINE ACTION RECOGNITION

In this section, we venture into action learning and recognition using the MSRAction3D Skeleton (MSRAS) dataset [31]. The MSRAS contains 20 actions performed by 10 different subjects repeating each action two to three times, recorded using a Kinect-like device. For each action, the skeletal data consists of a sequence of 3-dimensional coordinates and a confidence value (indicating how well each point was tracked) for 20 joints (examples shown in Fig 4). During our preliminary examination, several recordings were found to be missing significant chunks of the action sequence, presumably when the tracked skeleton was lost. As such, we removed recordings that had an average confidence below 55%[3]; of the initial 567 sequences, 544 (96%) were retained after this process.

For this problem, we used a generative modelling approach whereby each action class $a^i \in \mathcal{A}$ was represented by a separate OESGP model, $m^i \in \mathcal{M}$. Each model was trained to predict the velocity of each joint coordinate (20×3 outputs) given the current velocities (20×3 inputs). Inference was then performed using a Bayes filter to iteratively update a probability distribution over the model classes:

$$p(a_t^i|o, \mathcal{M}) = \frac{p(o_t|m^i)p(a_{t-1}^i)}{\sum_j p(o_t|m^j)p(a_{t-1}^j)} \tag{29}$$

where a_t^i is the action class i at time t and o is the observation (the velocities of the joints). We used a straightforward observational model where each coordinate was treated separately:

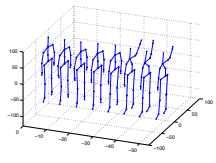
$$p(o_t|m^i) = \prod_k^{60} N(o_{k,t}|y_{k,t}^i, \sigma_{k,t}^i) \tag{30}$$

where $y_{k,t}^i$ and $\sigma_{k,t}^i$ are the k^{th} predicted mean and standard deviations using model m^i. The initial prior was assumed to be uniformly distributed and the most probable class at the end of the sequence was selected as the recognised action. In addition to being conceptually straightforward, this approach gives a probability distribution over the different actions during the entire course of the observed sequence. In our experiments, all OESGPs were initialised using the same parameters: $b = 0.5$, $\sigma = 0.05$ and capacity of 100 BVs with reservoirs of 100 neurons and 0.1 connectivity.

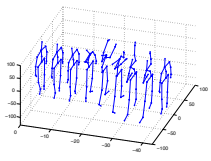
Following [31], we used three subsets of eight actions (shown in Fig. 4(e)) and conducted two separate experiments. For the "same-subject" experiment, training was conducted for each participant on trial 1 and 2, and tested on final trial. For the "cross-subject" experiment, training was conducted on all trials for subjects one through five and tested on recordings from subjects six through ten. Of the two, the cross-subject task is more challenging since different participants performed a given action in a dissimilar manner (See Figs. 4(c) and 4(d)).

Classification accuracies averaged over 10 runs are shown in Table IV. For comparison, we have included the scores obtained using the action graph method [31] (care should be taken when comparing these results since [31] used the *full* depth map version of the dataset instead of the tracked skeletons).

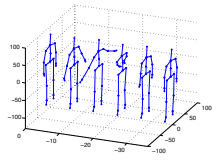
[3]At 50%, only 490 sequences were left after the removal and at 60%, too many erroneous sequences remained.



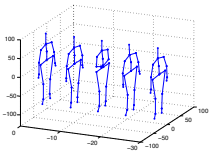
(a) S1: High Arm Wave



(b) S1: Tennis Serve



(c) S1: Hand Clap



(d) S5: Hand Clap

AS1	AS2	AS3
Horizontal arm wave	High arm wave	High throw
Hammer	Hand catch	Forward kick
Forward punch	Draw x	Side kick
High throw	Draw tick	Jogging
Hand clap	Draw circle	Tennis swing
Bend	Two hand wave	Tennis serve
Tennis serve	Forward kick	Golf swing
Pickup & throw	Side boxing	Pickup & throw

(e) Action Sets (AS) 1 to 3 [31]

Fig. 4. Four sequences from the MSRAction3D Skeletal dataset where examples 4(a)-4(c) were performed by subject one. In 4(d), note how the same action (hand clap) was performed in a different manner (without outstretched arms) by subject 5. Table 4(e) shows the three action subsets used in the study.

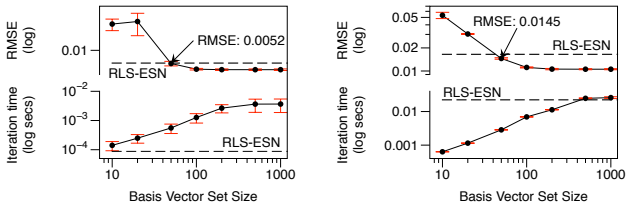
TABLE IV
ACCURACY SCORES FOR THE MSRACTION3D DATASET.

		Action Graph	OESGP
Same-Subject	AS1	0.934	0.950 (0.007)
	AS2	0.929	0.952 (0.012)
	AS3	0.963	0.965 (0.007)
	Overall	0.942	**0.955** (0.068)
Cross-Subject	AS1	0.729	0.806 (0.036)
	AS2	0.719	0.749 (0.064)
	AS3	0.792	0.871 (0.028)
	Overall	0.747	**0.809** (0.068)

The OESGP-based classifier achieved comparably high scores (above 95%) for the same-subject test, even without fine-tuning of parameters for individual action models. Moreover, we obtained substantially higher accuracies for the cross-subject test across all three action subsets (overall score of 80.9%). Since only a *single pass* was made through each sequence, training was rapid (requiring an average of 0.0006s per iteration). Predictions were more costly at 0.0089s per iteration since updates had to be made across all action classes, but well above the 30Hz required for real-time use. Therefore, models can be trained and used in real-time (e.g., for interactive robot-learning-by-demonstration scenarios) given reasonable limits on the number of action classes.

VI. DISCUSSION AND FUTURE WORK

In this section, we further investigate how OESGP's computational cost can be "regulated" by the size of the BV set and how this might affect the error. We also discuss possible future work; in particular, online parameter optimisation.



(a) Mackey-Glass (1D, 100 neurons) (b) Lorenz (3D, 1000 neurons)

Fig. 5. Computational cost of OESGP with varying basis vector set sizes.

A. Trading-off Computational Cost and Accuracy

Our experimental results demonstrate that the OESGP surpasses the other online methods in terms of lower error scores on the test problems, but this was generally achieved with a higher computational cost. This was not entirely surprising since OESGP produces variances which require additional computation time and is a global (sparse) method compared to LWPR, which uses smaller local models.

However, for fixed hyperparameters, the computational cost can be controlled by limiting the size of the basis vector set. Up to this point, our experiments used small reservoirs. Since the computational complexity for OESGP is $O(s_{\mathcal{B}}^2 + N_{\psi} s_{\mathcal{B}})$ per iteration, we see that for large reservoir sizes and small $s_{\mathcal{B}}$, OESGP may be cheaper than RLS-ESN, which takes $O(N_{\psi}^2)$. To illustrate this point, we trained OESGP using various sizes for the BV set, $s_{\mathcal{B}} = 10, 20, 50, 100, 200, 500, 1000$ on the Mackey-Glass and Lorenz problems (training and testing conducted throughout). For comparison, we used reservoirs of 100 and 1000 neurons for the Mackey-Glass and Lorenz problems respectively.

The RMSEs and iteration times (averaged over 10 independent runs) are shown in Fig. 5. As expected, we observed a positive relationship between the $s_{\mathcal{B}}$ and the average iteration time and an inverse relationship with the error. For the Mackey-Glass problem, to achieve an error rate comparable to that of RLS-ESN, $s_{\mathcal{B}} = 50$ where the computation time is $\approx 10^{-3}$s per iteration compared to $\approx 10^{-4}$ per iteration for RLS-ESN. On the Lorenz problem however, we find that at the same BV set size, $s_{\mathcal{B}} = 50$, OESGP (taking 0.0028s per iteration) not only obtains a lower mean error (0.0145) but is significantly faster than RLS-ESN (0.022s per iteration). Therefore, for larger reservoir sizes (e.g, the large hierarchical reservoirs of up to 20,000 neurons used in phoneme recognition [32]), OESGP may prove to be more efficient and accurate than RLS-ESN, while simultaneously providing uncertainties. That said, the error at different BV set sizes is problem-dependent; bounding it remains future work.

B. Online Hyperparameter and ESN Adaptation

The performance of RLS-ESN, SGD-ESN and OESGP are dependent on their parameters; for example, RLS-ESN relies on a proper choice of δ and λ; under suboptimal settings, convergence could be slow or yield suboptimal weights. Unlike LWPR, parameter adaptation is not currently a feature of any of online ESN training algorithms.

In the case of OESGP, accuracy depends on the chosen kernel and noise hyperparameters, as well as the maximum capacity of BV set. As discussed in the previous section, the maximum capacity would be determined by the computational requirements of the application. Theoretically, OESGP's hyperparameters can be optimised in a principled manner by maximising the log marginal likelihood, $\log p(\mathbf{y}|\mathcal{D}.\boldsymbol{\theta})$ In an online setting, we can perform this optimisation iteratively using stochastic gradient descent or evolutionary methods. However, there remain computational challenges to address. With each hyperparameter update, $\boldsymbol{\alpha}, \mathbf{C}$ and $\mathbf{Q}$ will have to be recomputed; this procedure may be prohibitively expensive for some applications. In addition, we have not touched upon the online adaptation of the ESN parameters, e.g., the reservoir size and spectral radius. Performing both ESN and parameter updates simultaneously remains an open problem.

VII. Conclusion

In this work, we presented the sparse online echo-state gaussian process (OESGP), a sparse Bayesian formulation of the echo-state network. In contrast to existing methods, the OESGP has the capability to *learn online*, *provide uncertainty estimates* and *model temporal dynamics*. To the best of our knowledge, this is the first algorithm possessing all three features. Experiments on a range of problems illustrate OESGP's effectiveness; it obtained significantly better error scores compared to existing online ESNs trained using RLS and SGD as well as the popular LWPR algorithm. As a case study, we used OESGP to create a highly-accurate Bayes classifier for an online action recognition. We anticipate that OESGP will be useful in a variety of applications areas where real-time learning is important. As examples, we are currently working on using OESGP in a hierarchical architecture [33] for robot-learning-by-demonstration and learning MR-POMDPs [34].

Acknowledgment

The authors thank members of the BioART Laboratory at Imperial College London, particularly Sotirios Chatzis and Dimitrios Korkinof for their help with ESGP. Harold Soh is supported by a Khazanah Global Scholarship. This work was partially funded by the EU FP7 ALIZ-E project (248116).

References

[1] F. Kianifard and W. H. Swallow, "A review of the development and application of recursive residuals in linear models," *Journal of the American Statistical Association*, vol. 91, no. 433, pp. 391–400, 1996.

[2] M. Lukosevicius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[3] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," German National Research Center for Information Technology, Bremen, Tech. Rep. 148, 2001.

[4] Z. Deng and Y. Zhang, "Collective behavior of a small-world recurrent neural system with scale-free distribution," *IEEE Trans. on Neural Networks*, vol. 18, no. 5, pp. 1364 –1375, Sept. 2007.

[5] A. Rodan and P. Tino, "Minimum complexity echo state network," *IEEE Trans. on Neural Networks*, vol. 22, no. 1, pp. 131 –144, jan. 2011.

[6] S. Vijayakumar, A. D'souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[7] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell, "Active learning with gaussian processes for object categorization," in *IEEE 11th International Conference on Computer Vision (ICCV)*, 2007, pp. 1–8.

[8] L. Csató and M. Opper, "Sparse on-line gaussian processes," *Neural Computation*, vol. 14, no. 3, pp. 641–668, March 2002.

[9] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," in *Advances in Neural Information Processing Systems*, 2003, pp. 593–600.

[10] P. Kountouriotis, D. Obradovic, S. Goh, and D. Mandic, "Multi-step forecasting using echo state networks," in *The International Conference on Computer as a Tool*, vol. 2. IEEE, 2005, pp. 1574–1577.

[11] S. Chatzis and Y. Demiris, "Echo state gaussian process," *IEEE Trans. on Neural Networks*, vol. 22, no. 9, pp. 1435 –1445, Sept. 2011.

[12] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Trans. on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.

[13] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[14] P. J. and Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Networks*, vol. 1, no. 4, pp. 339 – 356, 1988.

[15] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, June 1989.

[16] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[17] L. Csató, "Gaussian processes: iterative sparse approximations," Ph.D. dissertation, Aston University, 2002.

[18] J. Quiñonero Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *J. Mach. Learn. Res.*, vol. 6, pp. 1939–1959, December 2005.

[19] M. Opper, *A Bayesian approach to on-line learning*. Cambridge University Press, 1998, pp. 363–378.

[20] M. Seeger, C. Williams, and N. Lawrence, "Fast forward selection to speed up sparse gaussian process regression," in *Ninth Intl. Workshop on Artificial Intelligence and Statistics*, vol. 9, 2003.

[21] S. Keerthi and W. Chu, "A matching pursuit approach to sparse gaussian process regression," in *Advances in Neural Information Processing Systems 18*. Cambridge, MA: MIT Press, 2006, pp. 643–650.

[22] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org.

[23] D. Grollman, "Sparse online gaussian process c++ library." [Online]. Available: http://lasa.epfl.ch/ dang/code.shtml

[24] H. Soh, "Online Temporal Learning (OTL) C++ Library." [Online]. Available: http://www.haroldsoh.com/

[25] S. Vijayakumar, A. D'Souza, and S. Schaal, "LWPR: Locally Weighted Projection Regression library." [Online]. Available: http://www.ipab.inf.ed.ac.uk/slmc/software/lwpr/

[26] S. Klanke and S. Vijayakumar, "LWPR supplementary documentation," Tech. Rep., 2008.

[27] A. S. Weigend, *Time Series Prediction: Forecasting The Future And Understanding The Past*, ser. Sante Fe Institute Series, N. A. Gershenfeld, Ed. Westfield Press, Nov 1993.

[28] C. Richard, J. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. on Signal Processing*, vol. 57, no. 3, pp. 1058 –1067, March 2009.

[29] J. Kivinen, A. Smola, and R. Williamson, "Online learning with kernels," *IEEE Trans. on Signal Processing*, vol. 52, no. 8, pp. 2165 – 2176, 2004.

[30] V. Kadirkamanathan and M. Niranjan, "A function estimation approach to sequential learning with neural networks," *Neural Computation*, vol. 5, no. 6, pp. 954–975, Nov 1993.

[31] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2010, pp. 9–14.

[32] F. Triefenbach, A. Jalalvand, B. Schrauwen, and J.-P. Martens, "Phoneme recognition with large hierarchical reservoirs," in *Advances in Neural Information Processing Systems 23*, 2010, pp. 2307–2315.

[33] Y. Demiris and B. Khadhouri, "Hierarchical attentive multiple models for execution and recognition of actions," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 361–369, 2006.

[34] H. Soh and Y. Demiris, "Evolving policies for multi-reward partially observable markov decision processes (MR-POMDPs)," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, 2011, pp. 713–720.