Neural Networks ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2012 Special Issue

Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression

Arjan Gijsberts *, Giorgio Metta

Department of Robotics, Brain and Cognitive Sciences, Istituto Italiano di Tecnologia, Via Morego 30, 16163 Genoa, Italy

ARTICLE INFO

Keywords:
Incremental learning
Online learning
Function approximation
Real-time
Robotics

ABSTRACT

Novel applications in unstructured and non-stationary human environments require robots that learn from experience and adapt autonomously to changing conditions. Predictive models therefore not only need to be accurate, but should also be updated incrementally in real-time and require minimal human intervention. Incremental Sparse Spectrum Gaussian Process Regression is an algorithm that is targeted specifically for use in this context. Rather than developing a novel algorithm from the ground up, the method is based on the thoroughly studied Gaussian Process Regression algorithm, therefore ensuring a solid theoretical foundation. Non-linearity and a bounded update complexity are achieved simultaneously by means of a finite dimensional random feature mapping that approximates a kernel function. As a result, the computational cost for each update remains constant over time. Finally, algorithmic simplicity and support for automated hyperparameter optimization ensures convenience when employed in practice. Empirical validation on a number of synthetic and real-life learning problems confirms that the performance of Incremental Sparse Spectrum Gaussian Process Regression is superior with respect to the popular Locally Weighted Projection Regression, while computational requirements are found to be significantly lower. The method is therefore particularly suited for learning with real-time constraints or when computational resources are limited.

1. Introduction

The field of robotics is increasingly moving toward applications that require robots to act autonomously in unstructured human environments. The traditional approach to robotics, consisting of precisely characterizing the robot, its environment, and the desired task, is infeasible in this setting. Instead, in order to make accurate predictions of both the internal (i.e., the body) and external environment, the robot will be required to autonomously construct and maintain models by means of learning and adaptation. Machine learning techniques are obvious candidates to implement such predictive capabilities.

Despite significant advancements in statistical machine learning, automated construction and adaptation of non-linear predictive models are still far from trivial. In the described setting, an ideal learning algorithm needs not only to be accurate, but also to permit an indefinite number of incremental updates and to minimize the amount of human intervention required for successful operation. Furthermore, predictions have to be made in real-time to allow for responsive behavior and dynamic interactions with the environment. It is primarily this last requirement that poses difficulties for the majority of state of the art learning algorithms, since the computational cost of these methods tends to increase with the number of training samples. For instance, this increase is apparent for the popular class of kernel methods (KMs), since the model is described in terms of a kernel expansion over the training samples (Shawe-Taylor & Cristianini, 2004). Consequently, there will inevitably be a point in time at which the model fails to meet application specific timing requirements. It has to be noted that timeliness of predictions is not only important in the robotics domain, but also in fields such as industrial control, financial forecasting, or online advertising.

Gijsberts and Metta (2011) recently proposed Incremental Random Feature Ridge Regression (I-RFRR), an incremental learning algorithm intended specifically for use in an autonomous and real-time setting. Rather than counteracting the increase of the kernel expansion, as is commonly done in related methods, they avoid the problem altogether by approximating the kernel using a finite dimensional feature mapping (Rahimi & Recht, 2008a). Interestingly, this feature mapping is composed of multiple random mappings, and the approximation converges exponentially with the chosen number of random features. As a result, the computational requirements for an update only depend on the chosen number of random features, therefore satisfying

* Correspondence to: Idiap Research Institute, Rue Marconi 19, 1920 Martigny, Switzerland.
E-mail addresses: arjan.gijsberts@idiap.ch (A. Gijsberts), giorgio.metta@iit.it (G. Metta).

0893-6080/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2012.08.011

the predictability requirement for real-time applications as well as allowing accuracy to be traded for computational time. In this work, we present Incremental Sparse Spectrum Gaussian Process Regression (I-SSGPR), which applies similar techniques on top of the well-known Gaussian Process Regression (GPR) algorithm (Rasmussen & Williams, 2005). The close relationship to GPR has a number of important advantages, such as a strong theoretical foundation, an estimate of the predictive variance, and automated hyperparameter optimization by means of likelihood maximization. Although the separate components of this algorithm have been known in the machine learning community, their combination gives rise to an elegant and efficient learning method, as will be demonstrated with an extended empirical validation on a number of synthetic and real-world datasets.

In the following sections, we first present an overview of state of the art incremental learning methods in Section 2. The primary focus in this overview is on incremental KMs and techniques that have been presented specifically for use in robotics. The I-SSGPR algorithm is subsequently described in Section 3, which includes a detailed description of the finite dimensional kernel approximation and the efficient incremental update rule. This section ends with a discussion on its properties and merits with respect to related work. Section 4 contains comparative experiments on a number of synthetic and real robotics datasets, both in terms of generalization performance as well as computational efficiency. Finally, conclusions are given in Section 5.

2. Related work

A number of strategies have been proposed over the past decade to update KMs incrementally without the need to completely retrain the model. Initial efforts concentrated primarily on procedures that compute an exact update to a batch solution after adding (or removing) a training sample (e.g., Cauwenberghs & Poggio, 2001; Ma, Theiler, & Perkins, 2003; Martin, 2002). Clearly, the size of the resulting kernel expansion remains identical to the batch solution and the time complexity for each update is typically quadratic in the number of training samples. This high update complexity may be alleviated by computing approximate rather than exact updates. A popular algorithm that follows this approach is LASVM (Bordes, Ertekin, Weston, & Bottou, 2005), which can be shown to converge to the Support Vector Machine (SVM) solution when feeding the training samples multiple times. At each iteration, the algorithm seeks to add and remove samples from the active set by optimizing the coefficients of a pair of samples. Despite regular removal of samples from the active set, the size of the kernel expansion in LASVM still scales linearly with the number of training samples. Aside from being unsuitable in a real-time context, an additional difficulty with this method is that convergence requires independent and identically distributed (i.i.d.) sampling.

The problem of a linearly growing kernel expansion can be avoided by projecting ν-approximate linear dependent samples onto each other, where increasing the value ν induces more sparsity at the cost of accuracy. In Sparse Online Greedy ϵ-insensitive Support Vector Regression (ϵ-SVR) (Engel, Mannor, & Meir, 2002), this sparsification method is used in the context of an approximate SVM for regression (Vijayakumar & Wu, 1999), while Kernel Recursive Least Squares uses the same technique to incrementally compute a kernel-based least squares solution with restricted kernel expansion (Engel, Mannor, & Meir, 2004). A similar method is used in a Bayesian context by Csató and Opper (2002) to formulate Sparse Online Gaussian Process Regression, although here the linear dependence measure is subsequently multiplied with a likelihood-dependent term for the sample. Assuming a compact input space $\mathcal{X}$, it can be proved that the approximate linear dependence condition with $\nu > 0$ leads to a bounded kernel expansion (e.g., Engel et al., 2004). However, its exact size (and therefore computational requirements) is strongly data-dependent and cannot be computed a priori. Csató and Opper (2002) therefore impose in addition a strict upper bound on the size of the kernel expansion; samples are forcefully projected on the current active set once a predefined budget B has been reached. Even though this constitutes an effective upper bound on the computational requirements, there are two difficulties with this approach. First, the bound on the kernel expansion now depends both on the budget B and the approximation parameter ν, and both need to be tuned jointly using representative data to obtain an optimal tradeoff between computation and accuracy. Second, the additional budget constraint renders it difficult, if not impossible, to devise an approximation bound of the algorithm.

A large number of incremental methods have also been developed in the online learning[1] framework (Cesa-Bianchi & Lugosi, 2006; Vovk, Gammerman, & Shafer, 2005). The problem of increasing computational requirements is particularly prominent in this class of algorithms, as each update usually involves adding a sample to the kernel expansion. Several variants have therefore been proposed to limit the size of the kernel expansion, for instance by removing a sample at random (Cavallanti, Cesa-Bianchi, & Gentile, 2007), removing the oldest sample in the active set (Dekel, Shalev-Shwartz, & Singer, 2008; Kivinen, Smola, & Williamson, 2004), by projecting samples using approximate linear dependence (Orabona, Keshet, & Caputo, 2009), or based on the impact on an estimate of the misclassification rate (Crammer, Kandola, & Singer, 2004; Weston, Bordes, & Bottou, 2005). An important consideration when applying any of these algorithms is that low regret does not necessarily imply low risk, and so-called "online-to-batch" conversions are required to convert the sequence of predictors produced by the online algorithm into a single prediction function with low risk (Cesa-Bianchi & Gentile, 2008).

2.1. Methods proposed for robotics

The importance of incremental and computationally efficient regression methods for robotics has led to the development of a number of algorithms targeted specifically for use in this application domain (see Sigaud, Salaün, & Padois, 2011, for a survey). One of the first methods that gained widespread acceptance is Receptive Field Weighted Regression (RFWR), proposed by Schaal and Atkeson (1998). RFWR is an incremental Locally Weighted Regression (LWR) variant, in which the function under consideration is approximated by piecewise linear models. The contribution of each of these local models (i.e., the receptive fields) to the overall prediction is weighed such that they are active only within a localized region of the input space. New receptive fields are allocated as needed if existing fields are not activated sufficiently. The number of receptive fields that will be allocated therefore depends on a minimum activation threshold a_{gen}, the input space $\mathcal{X}$, and the smoothness of the function generating the data. It follows from the well-known curse of dimensionality that the number of receptive fields scales exponentially with the input dimensionality. This exponential increase is alleviated to

[1] Online learning has become synonymous with learning in the regret minimization framework. To avoid confusion, the term "online" will therefore strictly refer to algorithms developed and analyzed within this framework, while the term "incremental" will be used in a more broad sense for algorithms that can process additional training samples with having to recompute the solution.

Please cite this article in press as: Gijsberts, A., & Metta, G. Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression. Neural Networks (2012), doi:10.1016/j.neunet.2012.08.011

some extent in Locally Weighted Projection Regression (LWPR) (Vijayakumar, D'souza, & Schaal, 2005), which uses Partial Least Squares in the local models to reduce the impact of redundant and irrelevant input dimensions. This improvement has led LWPR to gain significant popularity for use in incremental robotics learning problems.

It is relevant to stress the fundamental difference between LWPR and KMs: the latter methods perform global linear regression in an implicit high-dimensional feature space, whereas the former method is a combination of multiple linear regression models that operate individually within a (small) region of the original input space. An intrinsic property of LWPR is thus that learning takes place at two levels, namely (1) finding an appropriate structuring of the input space and (2) fitting linear models to each subspace. This interplay between both levels of learning is governed using a large number of hyperparameters, among which are the initial distance matrix, the initial learning rate and meta learning rate, activation thresholds for receptive field creation and pruning, a projection threshold for PLS, and a penalty term preventing infinite shrinkage of the receptive fields. When configured correctly, an advantage of local learning is that efficiency is improved by modeling only relevant regions in the input space. However, it is notoriously difficult to find a satisfactory configuration, and the necessity to learn both the local models and their structure at the same time negatively affects the sample complexity. It is therefore common that LWPR requires a much larger number of training samples to achieve performance similar to KMs. Nguyen-Tuong, Seeger, and Peters (2009) propose Local Gaussian Processes (LGP) in an attempt to improve the accuracy of LWPR by replacing the linear models with GPR models. Fewer receptive fields are necessary in this approach, as GPR models are more complex than linear models and can therefore accurately model a larger region of the input space. A nearly identical approach has been proposed by Schneider and Ertel (2010), the primary difference being that they employ a variant of GPR with a heteroskedastic noise model (Kersting, Plagemann, Pfaff, & Burgard, 2007). Nonetheless, the performance of both approaches is still inferior with respect to a global GPR, and the only advantage is a reduction in computational requirements. Surprisingly, for methods proposed explicitly for incremental and real-time learning, the computational complexity in all these methods is typically described in subjective terms (e.g., "efficient" or "fast") rather than formal terms (e.g., asymptotic complexity or an explicit upper bound). It is important to realize that localization does not necessarily decrease complexity, as the problem of an increasing kernel expansion is not fundamentally different from an increasing number of local models. Furthermore, there is little theoretical justification for these methods and there are no known generalization or regret bounds.

3. Algorithm

The common aspect of most of the methods identified in the previous section is that growth of the kernel expansion is restricted using an additional sparsification procedure on top of a standard incremental KM. This procedure typically introduces a significant computational overhead and additional hyperparameters that may not be trivial to tune (e.g., the measure of linear dependence ν). Furthermore, it is often difficult to exactly quantify their computational cost, which is inconvenient when employed in a strict real-time environment. This can be circumvented by imposing a fixed budget on the kernel expansion, though this requires an additional selection procedure to explicitly remove training data.

Here we implement an alternative approach to avoid these drawbacks, based on the realization that the kernel trick and the representer theorem are at the core of the problem. As noted by Shawe-Taylor and Cristianini (2004), a key property that is required of a kernel function for an application is that its evaluation should require less computation than would be needed for explicit evaluation of the corresponding feature mapping. This is unlikely in the case of a large or potentially infinite number of training samples. It is therefore computationally advantageous to avoid the kernel machinery and its consequences by performing the feature mapping explicitly. Unfortunately, for some kernels, among which is the Radial Basis Function (RBF) kernel, this feature mapping is infinite dimensional.

In order to resolve this issue, we apply a finite dimensional random feature mapping that approximates shift invariant kernel functions (Rahimi & Recht, 2008a). Though this technique has originally been proposed for large-scale batch learning, Gijsberts and Metta (2011) have combined it with an efficient update rule to obtain an incremental variant of Ridge Regression (RR) with constant update complexity. Experimental validation has demonstrated that this learning algorithm compares favorably to competing algorithms both in terms of generalization as well as computational performance (Droniou, Ivaldi, Stalph, Butz, & Sigaud, 2012; Gijsberts & Metta, 2011). Its performance, however, depends critically on the chosen hyperparameter configuration. Here we address the difficulty of tuning these parameters by employing the techniques in a GPR setting to obtain an incremental variant of the Sparse Spectrum Gaussian Process Regression (SSGPR) (Lázaro-Gredilla, Quiñonero-Candela, Rasmussen, & Figueiras-Vidal, 2010). As a result, all hyperparameters can be optimized in a principled manner using gradient ascent of the log marginal likelihood. Moreover, as an additional benefit the GPR-based algorithm also provides an estimate of the predictive variance.

3.1. Standard Bayesian or linear Gaussian process regression

The starting point of our algorithm is given by standard linear Bayesian regression (for a more thorough treatment, see Rasmussen & Williams, 2005). In this model, it is assumed that the outputs can be described by a linear model, where the observed outputs are corrupted by i.i.d. Gaussian noise. We can thus write

$$y = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \epsilon,$$

where $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ and $\epsilon \sim \mathcal{N}\left(0, \sigma_n^2\right)$. Furthermore, it is assumed that the outputs y are zero-mean.[2] The likelihood of the vector of observed outputs $\boldsymbol{y} = [y_1, \ldots, y_m]^{\mathrm{T}}$ given the $m \times n$ matrix of input samples $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m]^{\mathrm{T}}$ and the weight vector $\boldsymbol{w}$ is then

$$p(\boldsymbol{y}|\boldsymbol{w}, \boldsymbol{X}) = \mathcal{N}\left(\boldsymbol{X}\boldsymbol{w}, \sigma_n^2 \boldsymbol{I}\right).$$

Furthermore, let us define a prior over the weight vector $\boldsymbol{w}$ as a multi-variate Gaussian with zero mean and covariance matrix $\boldsymbol{\Sigma}_p$, such that $\boldsymbol{w} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\Sigma_p}\right)$. In the Bayesian framework, inference is based on the posterior distribution over the weight vector $\boldsymbol{w}$, given the observations $\boldsymbol{X}$ and $\boldsymbol{y}$. Applying Bayes' rule on the prior and likelihood, we obtain the posterior

$$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{A}^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}, \sigma_n^2 \boldsymbol{A}^{-1}\right), \tag{1}$$

where the covariance matrix $\boldsymbol{A} = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \sigma_n^2 \boldsymbol{\Sigma}_p^{-1}$. Model predictions are obtained by integrating over all possible $\boldsymbol{w}$ with respect to their posterior distribution in Eq. (1). This results in the predictive distribution

$$p(y|\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y},\ \sigma_n^2\left(1 + \boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}^{-1}\boldsymbol{x}\right)\right), \tag{2}$$

[2] This can be ensured by subtracting the sample mean or, alternatively, by appending an additional constant bias term in the input representation.

Please cite this article in press as: Gijsberts, A., & Metta, G. Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression. Neural Networks (2012), doi:10.1016/j.neunet.2012.08.011

where $(\boldsymbol{x}, y)$ is an individual test sample, as opposed to the training data $(\boldsymbol{X}, \boldsymbol{y})$. In addition, note that the additional σ_n^2 in the variance is necessary to compensate for the noise present in the test samples.

Generalization of Bayesian linear regression to non-linear problems involves replacing $\boldsymbol{x}$ with a non-linear feature map $\phi(\boldsymbol{x})$. Applying this substitution in (2), we obtain the predictive posterior

$$p(y|\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}\left(\phi(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{A}^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{y},\ \sigma_n^2 \left(1 + \phi(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{A}^{-1} \phi(\boldsymbol{x})\right)\right), \tag{3}$$

where $\boldsymbol{\Phi} = \phi(\boldsymbol{X})$ and

$$\boldsymbol{A} = \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} + \sigma_n^2 \boldsymbol{\Sigma}_p^{-1}. \tag{4}$$

At this point, the well-known kernelized GPR can be obtained by defining a kernel (or covariance) function

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left\langle \boldsymbol{\Sigma}_p^{\frac{1}{2}} \phi(\boldsymbol{x}_i)\ ,\ \boldsymbol{\Sigma}_p^{\frac{1}{2}} \phi(\boldsymbol{x}_j) \right\rangle,$$

where $\boldsymbol{\Sigma}_p^{\frac{1}{2}}$ is the square root of positive definite matrix $\boldsymbol{\Sigma}_p$ (denoted as $\boldsymbol{\Sigma}_p \succ 0$). The posterior distribution in Eq. (3) can then equivalently be written as

$$p(y|\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{k}^{\mathrm{T}} \boldsymbol{G}^{-1} \boldsymbol{y},\ k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}^{\mathrm{T}} \boldsymbol{G}^{-1} \boldsymbol{k} + \sigma_n^2\right),$$

with $\boldsymbol{K} = \left[k(\boldsymbol{x}_i, \boldsymbol{x}_j)\right]_{i,j=1}^{m}$, $\boldsymbol{k} = [k(\boldsymbol{x}, \boldsymbol{x}_i)]_{i=1}^{m}$, and $\boldsymbol{G} = \boldsymbol{K} + \sigma_n^2 \boldsymbol{I}$. However, as argued previously, computing the inverse of an $m \times m$ matrix $\boldsymbol{G}$ is significantly more costly than computing the inverse of $\boldsymbol{A}$ when $m \gg n$. Furthermore, extending the solution with an additional sample implies increasing the dimensionality of $\boldsymbol{G}$ and the size of the kernel expansion.

3.2. Sparse spectrum Gaussian process regression

The kernel expansion is only strictly necessary in the absence of a finite dimensional representation of the feature space of a given kernel. If such a representation exists, however, it can be advantageous to avoid the kernel machinery by explicitly mapping the samples in the feature space. Unfortunately, the feature mapping for the popular RBF kernel is infinite dimensional and can thus not be explicitly computed. Rahimi and Recht (2008a), however, demonstrate that the RBF and other shift invariant kernels $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = k(\boldsymbol{x}_i - \boldsymbol{x}_j)$ can be approximated to an arbitrary precision using a finite dimensional random feature mapping. Their approach relies on Bochner's theorem, which states that any positive definite and continuous function is the Fourier transform of a finite positive Borel measure (Bochner, 1933). Since shift invariant kernel functions are positive definite, these can be described as the Fourier transform of a unique measure μ. Consequently, if μ is scaled to integrate to 1, then the kernel can be estimated by sampling features from the spectral domain according to μ (e.g., Shawe-Taylor & Cristianini, 2004, Chapter 9.7). We can thus write

$$k(\boldsymbol{x}_i - \boldsymbol{x}_j) = \int_{\mathbb{R}^n} e^{-i\boldsymbol{\omega}^{\mathrm{T}}(\boldsymbol{x}_i - \boldsymbol{x}_j)} \mu(\boldsymbol{\omega})\ \mathrm{d}\boldsymbol{\omega} = \mathbb{E}_{\boldsymbol{\omega}}\left[z_{\boldsymbol{\omega}}(\boldsymbol{x}_i)^{\mathrm{T}}\, z_{\boldsymbol{\omega}}(\boldsymbol{x}_j)\right],$$

where

$$z_{\boldsymbol{\omega}}(\boldsymbol{x}) = \left[\cos(\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x})\ ,\ \sin(\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x})\right]^{\mathrm{T}}.$$

In other words, the inner product $\left\langle z_{\boldsymbol{\omega}}(\boldsymbol{x}_i)\ ,\ z_{\boldsymbol{\omega}}(\boldsymbol{x}_j)\right\rangle$ gives an unbiased estimate of any shift invariant kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, assuming that the spectral frequency $\boldsymbol{\omega}$ is drawn according to the measure μ. Given a kernel function, the corresponding probability density function can be obtained by computing its inverse Fourier transform and properly scaling the measure. For instance, the probability density function for the isotropic RBF kernel is Gaussian and it suffices to sample $\boldsymbol{\omega} \sim \mathcal{N}(\boldsymbol{0}, 2\gamma \boldsymbol{I})$.

The variance of the approximation can be lowered arbitrarily by drawing multiple $\boldsymbol{\omega} \sim \mu$ and averaging over the corresponding features $z_{\boldsymbol{\omega}}$, since each feature computes an unbiased random sample of the kernel k. Concatenating D random features $z_{\boldsymbol{\omega}}$ and normalizing the result, we obtain the feature mapping $\phi : \mathcal{X} \mapsto \mathbb{R}^{2D}$ given by

$$\phi(\boldsymbol{x}) = \frac{1}{\sqrt{D}} \left[z_{\boldsymbol{\omega}_1}(\boldsymbol{x})^{\mathrm{T}}, \ldots, z_{\boldsymbol{\omega}_D}(\boldsymbol{x})^{\mathrm{T}}\right]^{\mathrm{T}}. \tag{5}$$

It is clear that $k(\boldsymbol{x}_i, \boldsymbol{x}_j) \approx \left\langle \phi(\boldsymbol{x}_i)\ ,\ \phi(\boldsymbol{x}_j)\right\rangle$ for sufficiently large D and $\boldsymbol{\omega}_d \sim \mu$ for $1 \leq d \leq D$.

Approximation of the kernel (or covariance) function using random features has been investigated recently[3] in the context of GPR by Lázaro-Gredilla et al. (2010) under the name SSGPR. This involves applying the explicit random feature mapping ϕ from Eq. (5) in the formulation of the linear GPR, as was demonstrated in Eq. (3). In the following, we will constrain our attention to the popular Asymmetric Squared Exponential (ASE) kernel function (also known as the anisotropic RBF kernel). This kernel is given by

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_f^2\, e^{-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{x}_j)^{\mathrm{T}} \boldsymbol{M} (\boldsymbol{x}_i - \boldsymbol{x}_j)}$$

where $\boldsymbol{M}$ is an $n \times n$ diagonal matrix with $\boldsymbol{M}_{ii} = \ell_i^{-2}$ for $1 \leq i \leq n$ and factor σ_f denotes the signal variance. The characteristic length scales $\boldsymbol{\ell} > \boldsymbol{0}$ describe both the overall bandwidth and the relative importance of each input dimension, and may even be used to cancel out irrelevant dimensions (i.e., $\ell_i \to \infty$). Optimization of these length scales therefore constitutes a form of Automatic Relevance Detection (ARD) (MacKay, 1995). The random feature mapping that corresponds to this kernel is

$$\phi(\boldsymbol{x}) = \frac{\sigma_f}{\sqrt{D}} \left[\sin(\boldsymbol{\omega}_1^{\mathrm{T}}\boldsymbol{x})\ ,\ \cos(\boldsymbol{\omega}_1^{\mathrm{T}}\boldsymbol{x})\ ,\ \ldots,\ \sin(\boldsymbol{\omega}_D^{\mathrm{T}}\boldsymbol{x})\ ,\ \cos(\boldsymbol{\omega}_D^{\mathrm{T}}\boldsymbol{x})\right], \tag{6}$$

with the frequencies $\boldsymbol{\omega} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{M})$. Since $\boldsymbol{M}$ is diagonal, it suffices to draw $\boldsymbol{\omega}$ from $\mathcal{N}(0, 1)$ and subsequently scale these according to $\boldsymbol{\ell}$.

A consequence of the ASE kernel is that the number of kernel parameters increases from 1 to n. Fortunately, a favorable property of GPR compared to other KMs is that it permits efficient optimization of the hyperparameters using the marginal likelihood (Rasmussen & Williams, 2005). In case of SSGPR, the negative log marginal likelihood is given by

$$-\ln p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \frac{1}{2\sigma_n^2}\left(\boldsymbol{y}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{y}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{A}^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}\right) + \frac{1}{2}\ln\det\boldsymbol{A} - \frac{D}{2}\ln\sigma_n^2 + \frac{m}{2}\ln 2\pi\sigma_n^2,$$

where $\boldsymbol{A}$ is as defined in Eq. (4) and $\boldsymbol{\theta}$ is a vector containing all hyperparameters. Interestingly, considering the spectral frequencies $\boldsymbol{\omega}_d$ for $1 \leq d \leq D$ as parameters rather than constants, these can be optimized using the log marginal likelihood as well (Lázaro-Gredilla et al., 2010). Noting that the distribution of the spectral frequencies is uniquely related to the kernel via the measure μ, it is clear that optimizing these frequencies equates to tuning the kernel function to the data. Although this may result in highly efficient and sparse models, there are two disadvantages to optimization of

[3] An earlier technical report seems to suggest that Lázaro-Gredilla, Quiñonero-Candela, and Figueiras-Vidal (2007) discovered this technique independently from Rahimi and Recht.

Please cite this article in press as: Gijsberts, A., & Metta, G. Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression. Neural Networks (2012), doi:10.1016/j.neunet.2012.08.011

the frequencies. First, the resulting feature mapping will no longer be an approximation of the chosen kernel, therefore invalidating theoretical results that depend on this relationship between mapping and kernel. Second, the number of tunable hyperparameters increases from $n + 2$ (in case of the ASE kernel) to $n + 2 + Dn$, therefore drastically increasing the risk of overfitting.

3.3. Efficient incremental updates

A useful property of the linear variant of GPR in the scope of this work is that it allows efficient and exact updates of the solution using well-known linear algebra methods. Here we will demonstrate a numerically stable and efficient technique to obtain an incremental (or online) variant of the SSGPR algorithm, which we will subsequently refer to as I-SSGPR. Notice that the optimal solution from Eq. (3) can be decomposed in terms of the covariance matrix $\boldsymbol{A}$ and a vector $\boldsymbol{b} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}$. Furthermore, arrival of a new sample $(\boldsymbol{x}_t, y_t)$ at time step t can be described as $\boldsymbol{\Phi}_t = [\boldsymbol{\Phi}_{t-1}^{\mathrm{T}}, \boldsymbol{\phi}_t]^{\mathrm{T}}$ and $\boldsymbol{y}_t = [\boldsymbol{y}_{t-1}^{\mathrm{T}}, y_t]^{\mathrm{T}}$, where we defined $\boldsymbol{\phi}_t = \phi(\boldsymbol{x}_t)$ for convenience. Applying these update rules, the matrix $\boldsymbol{A}$ can be formulated recursively as

$$\begin{aligned}\boldsymbol{A}_t &= \sigma_n^2\boldsymbol{\Sigma}_p^{-1} + \boldsymbol{\Phi}_t^{\mathrm{T}}\boldsymbol{\Phi}_t = \sigma_n^2\boldsymbol{\Sigma}_p^{-1} + \boldsymbol{\Phi}_{t-1}^{\mathrm{T}}\boldsymbol{\Phi}_{t-1} + \boldsymbol{\phi}_t\boldsymbol{\phi}_t^{\mathrm{T}} \\ &= \boldsymbol{A}_{t-1} + \boldsymbol{\phi}_t\boldsymbol{\phi}_t^{\mathrm{T}},\end{aligned}$$

from which we can conclude that each additional training sample constitutes a rank-1 update to matrix $\boldsymbol{A}_{t-1}$. Similarly, vector $\boldsymbol{b}$ can be updated recursively as

$$\boldsymbol{b}_t = \boldsymbol{\Phi}_t^{\mathrm{T}}\boldsymbol{y}_t = \boldsymbol{\Phi}_{t-1}^{\mathrm{T}}\boldsymbol{y}_{t-1} + \boldsymbol{\phi}_t y_t = \boldsymbol{b}_{t-1} + \boldsymbol{\phi}_t y_t.$$

The recursion is complete by setting the initial configuration $\boldsymbol{A}_0 = \sigma_n^2\boldsymbol{\Sigma}_p^{-1}$ and $\boldsymbol{b}_0 = \boldsymbol{0}$. It is therefore no longer necessary to store previous samples in memory, since only $\boldsymbol{A}_t$ and $\boldsymbol{b}_t$ are required at each time step. However, this formulation requires explicit inversion of $\boldsymbol{A}_t$ at each update, causing the time complexity to be in $\mathcal{O}(D^3)$ and thus independent of the number of training samples.

This time complexity can be reduced to $\mathcal{O}(D^2)$ by directly updating the inverse of $\boldsymbol{A}$ using the Sherman–Morrison formula (Golub & Loan, 1996). The simplicity and recursive nature of this update procedure has made it very popular for many applications. However, a disadvantage of explicitly updating the inverse matrix is sensitivity to roundoff errors (e.g., Björck, 1996, Section 3.1). A numerically more stable alternative is to update the Cholesky factor of $\boldsymbol{A}$ instead (Björck, 1996; Golub & Loan, 1996), known as the QR algorithm in the field of adaptive filtering (Sayed, 2008, Section 35.2). Let us define the upper triangular Cholesky factor $\boldsymbol{R}_t$ such that $\boldsymbol{A}_t = \boldsymbol{R}_t^{\mathrm{T}}\boldsymbol{R}_t$ for each time step t. Note that $\boldsymbol{R}_t$ is full rank and guaranteed to be unique, since $\boldsymbol{\Sigma}_p \succ 0$ implies that $\boldsymbol{A}_t \succ 0$ for each t. Following earlier notation, we are interested in the rank-1 update problem

$$\boldsymbol{R}_t^{\mathrm{T}}\boldsymbol{R}_t = \boldsymbol{A}_t = \boldsymbol{A}_{t-1} + \boldsymbol{\phi}_t\boldsymbol{\phi}_t^{\mathrm{T}} = \boldsymbol{R}_{t-1}^{\mathrm{T}}\boldsymbol{R}_{t-1} + \boldsymbol{\phi}_t\boldsymbol{\phi}_t^{\mathrm{T}},$$

where the initial matrix $\boldsymbol{R}_0$ is the Cholesky factor of $\sigma_n^2\boldsymbol{\Sigma}_p^{-1}$. In the following, we will assume $\boldsymbol{\Sigma}_p = \boldsymbol{I}$ and thus $\boldsymbol{R}_0 = \sigma_n\boldsymbol{I}$. This rank-1 update can be computed by reformulating the problem as

$$\boldsymbol{R}_t^{\mathrm{T}}\boldsymbol{R}_t = \boldsymbol{R}_{t-1}^{\mathrm{T}}\boldsymbol{R}_{t-1} + \boldsymbol{\phi}_t\boldsymbol{\phi}_t^{\mathrm{T}} = \tilde{\boldsymbol{R}}_t^{\mathrm{T}}\tilde{\boldsymbol{R}}_t,$$

where the temporary $(D + 1) \times D$ matrix $\tilde{\boldsymbol{R}}_t = \left[\boldsymbol{R}_{t-1}^{\mathrm{T}}, \boldsymbol{\phi}_t\right]^{\mathrm{T}}$. It follows that the updated Cholesky factor $\boldsymbol{R}_t$ can be obtained by introducing zeros in the last row of $\tilde{\boldsymbol{R}}_t$ by means of D Givens rotations (Björck, 1996; Golub & Loan, 1996). Eliminating the bottom row in this manner, the remaining top $D \times D$ block will contain the updated Cholesky factor $\boldsymbol{R}_t$. Subsequently, $\boldsymbol{R}_t$ can be used to compute the predictive mean and variance (see Eq. (3)) by means of back and forward substitution. Although this procedure might seem elaborate, the time and space complexity are in fact $\mathcal{O}(D^2)$ and thus identical to rank-1 updates of the inverse matrix. The computational cost is also similar in an absolute sense, such that incremental updating of the Cholesky factor is to be preferred in practice due to its increased numerical stability.

Table 1
Typical operation counts for the update procedure of I-SSGPR as described in Algorithm 1.

Line	$*/\div$	$+/-$	sin / cos	$\sqrt{\ }$	abs
6	$Dn + 2D + 1$	$D(n-1)$	$2D$		
7	$2D$	$2D - 1$			
8	$4D^2$	$4D^2 - 2D$			
9	$2D + 1$	$2D$			
10	$2D$	$2D$			
11	$8D^2 + 8D$	$4D^2$		$2D$	$4D$
12	$4D^2 + 2D$	$4D^2 - 2D$			
	$16D^2 + D(n + 18) + 2$	$12D^2 + D(n + 1) - 1$	$2D$	$2D$	$4D$

3.4. Discussion

The I-SSGPR algorithm is fundamentally different from methods described earlier in Section 2. In these methods, sparsity is achieved by approximating the solution to a standard KM optimization problem, whereas I-SSGPR approximates the kernel function and therefore the optimization problem itself rather than the solution. Although the difference is subtle, this has significant theoretical and practical consequences. First, the random feature mapping can be considered a kernel function in its own right and it follows that I-SSGPR is a standard KM (albeit with a non-conventional kernel). As such, a wealth of theoretical results and extensions for both linear Bayesian regression and kernelized GPR are directly applicable to the method described here (e.g., Zhdanov & Kalnishkan, 2010). Second, due to the finiteness of the feature mapping, it is no longer required to incorporate additional control mechanisms to contain the growth of the kernel expansion. In other words, rather than dealing with the symptoms of an increasing kernel expansion, these are avoided altogether. This simplifies the algorithm drastically and causes the computational and space complexity for an update to be in $\mathcal{O}(1)$ with respect to the number of training samples. Perhaps more important than this constant asymptotic growth rate, the absolute number of operations required for each update also remains constant over time (see Table 1). This data-independent predictability of the computational requirements make I-SSGPR particularly suitable for use in hard real-time applications.

The pseudocode for I-SSGPR is demonstrated in Algorithm 1, where we notice the simplicity as compared to competing algorithms. Arguably, complicated algorithms are harder to comprehend, implement, and apply in practice. This holds in particular for LWPR, which is often found to be cumbersome in practical use due to the difficulty of tuning a large number of non-intuitive hyperparameters. In contrast, I-SSGPR is easy to implement using standard linear algebra routines and it improves over I-RFRR by allowing hyperparameters to be tuned automatically using log marginal likelihood optimization. Furthermore, the number of random features D is the only parameter responsible for regulating the tradeoff between predictive accuracy and computational cost. The maximum deviation between the feature mapped inner product and the kernel converges monotonically as $\mathcal{O}\left(\frac{1}{\sqrt{D}}\right)$ for each $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^n$ (Rahimi & Recht, 2008b, Theorem 3.2), as confirmed empirically in Fig. 1. Consequently, this parameter effectively allows one to maximize the approximation accuracy given domain specific timing constraints and available resources. Since the incremental updates are exact, it follows that I-SSGPR produces at each time step a close

Please cite this article in press as: Gijsberts, A., & Metta, G. Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression. Neural Networks (2012), doi:10.1016/j.neunet.2012.08.011

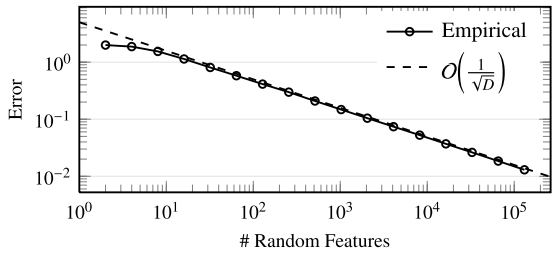

Fig. 1. Convergence of the maximum error between k and $\langle \phi, \phi \rangle$ over a set of samples with respect to the number of random features D. Note the log scale on both axes.

Algorithm 1 I-SSGPR with ASE kernel

Require: $\sigma_n > 0, \sigma_f > 0, \boldsymbol{M} \succ 0, D > 0$
$\boldsymbol{R} \leftarrow \sigma_n \boldsymbol{I}_{2D \times 2D}$
$\boldsymbol{w} \leftarrow \boldsymbol{0}_{2D \times 1}$
$\boldsymbol{b} \leftarrow \boldsymbol{0}_{2D \times 1}$
$\Omega \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{M})_{D \times n}$
for all $(\boldsymbol{x}, y)$ **do**
  $\boldsymbol{\phi} \leftarrow \frac{\sigma_f}{\sqrt{D}} \left[\cos(\Omega \boldsymbol{x})^{\mathrm{T}}, \sin(\Omega \boldsymbol{x})^{\mathrm{T}}\right]^{\mathrm{T}}$ // *Eq. (6)*
  $\hat{y} \leftarrow \langle \boldsymbol{w}, \boldsymbol{\phi} \rangle$
  $\boldsymbol{v} \leftarrow \boldsymbol{R}^{\mathrm{T}} \backslash \boldsymbol{\phi}$
  $s^2 \leftarrow \sigma_n^2 \left(1 + \langle \boldsymbol{v}, \boldsymbol{v} \rangle\right)$
  $\boldsymbol{b} \leftarrow \boldsymbol{b} + \boldsymbol{\phi} y$
  $\boldsymbol{R} \leftarrow \textsc{CholeskyUpdate}(\boldsymbol{R}, \boldsymbol{\phi})$ // *Section 3.3*
  $\boldsymbol{w} \leftarrow \boldsymbol{R} \backslash \left(\boldsymbol{R}^{\mathrm{T}} \backslash \boldsymbol{b}\right)$
  yield $\left(\hat{y}, s^2\right)$
end for

approximation of batch GPR trained on all samples observed thus far. It is therefore not necessary to perform multiple passes over the samples and, unlike incremental methods such as stochastic gradient descent, the ordering of the samples is irrelevant for the final solution. This property makes the algorithm relatively resistant against violation of the i.i.d. sampling assumption.

4. Empirical evaluation

This section presents an empirical evaluation of the proposed I-SSGPR algorithm in terms of generalization performance, computational requirements, and practicality. The focus in these experiments is a comparison with LWPR and GPR on a range of synthetic and real-life datasets. The former method has been used extensively for incremental learning tasks — most prominently in the robotics community — and it is of interest to examine whether I-SSGPR can be considered a viable alternative. In contrast, the comparison with GPR serves as a reference, since I-SSGPR approximates a continuously updated kernel GPR. This comparison also allows us to investigate the advantage of incremental learning over batch solutions in a realistic learning setting. We expect that incremental learning may have a significant advantage, since the assumption that training and test data are drawn from the same distribution is typically violated in such settings.

All results reported in this section have been obtained using software written primarily in Python. Noteworthy details include that we used a thin Python wrapper around the "official" LWPR implementation (Klanke, Vijayakumar, & Schaal, 2008), that rank-1 Cholesky updates were computed using LINPACK's `dchud` function (Dongarra, Bunch, Moler, & Stewart, 1979), and that kernel functions were evaluated by means of an extension written in C. Though not used for the present experiments, a C++ implementation of I-SSGPR is publicly available as part of the RobotCub repository.[4]

4.1. Synthetic data

The first experiment evaluates SSGPR on the synthetic Cross datasets, previously used by Vijayakumar et al. (2005) to evaluate the generalization performance and scaling behavior of LWPR. In the base Cross 2D dataset, samples are generated according to the two-dimensional function

$$f(\boldsymbol{x}) = \max\left\{e^{-10\boldsymbol{x}_1^2}, e^{-50\boldsymbol{x}_2^2}, 1.25e^{-5\left(\boldsymbol{x}_1^2+\boldsymbol{x}_2^2\right)}\right\}.$$

This function is characterized by a mixture of areas of both low and high curvature. As argued by Vijayakumar et al., low complexity algorithms have difficulties to accurately capture the non-linearities in high curvature regions, while high complexity models may overfit on the regions of low curvature. In Cross 10D, eight additional input features are added to the input space by applying a random 10-dimensional rotation matrix, effectively causing a high degree of redundancy in the input space. For the third variant (Cross 20D), the input space of Cross 10D is extended with ten irrelevant input features, each of which consists of $\mathcal{N}\left(0, 0.05^2\right)$ noise. In all three cases, the training data consists of 500 randomly drawn samples corrupted by $\mathcal{N}\left(0, 0.1^2\right)$ random noise. The 1681 test samples, instead, are noiseless and distributed uniformly on a 41×41 grid in the unit square of the input space.

The synthetic datasets are used as a benchmark to compare SSGPR with LWPR and GPR, where we note that batch SSGPR is identical to performing a single pass of I-SSGPR over the training data.[5] For LWPR, the initial distance metric *init_D*, learning rate *init_alpha*, and meta learning rate *meta_rate* hyperparameters are optimized greedily in this respective order using a derivative of the procedure described in the supplementary documentation.[6] GPR, on the other hand, is employed with the ASE kernel function and its hyperparameters are optimized using log marginal likelihood maximization. SSGPR is used in a nearly identical setup, the only difference being the option to optimize the spectral frequencies as well. If chosen, then the number of tunable parameters increases drastically with an additional $D \times n$ hyperparameters. The number of random sparse spectrum features D is configured by the user and not subject to optimization.

The results shown in Fig. 2 demonstrate good generalization performance for GPR, achieving a normalized Mean Squared Error (nMSE) of approximately 0.02 on all three datasets. We can conclude that GPR with ASE kernel remains effective also in the case of redundant and irrelevant input features. As expected in case of SSGPR with "fixed" spectral frequencies, increasing the number of sparse spectrum features improves the approximation and therefore the generalization performance in nearly all cases. Interestingly, the method requires more random features to obtain satisfactory performance with Cross 2D than with the higher dimensional variants. This may be explained by the small number of tunable hyperparameters in the two-dimensional case, which makes it harder to effectively tune the model to the data. Increasing

[4] See the *learningMachine* library in the RobotCub framework, available at http://eris.liralab.it/iCub/.

[5] Rounding errors due to working in finite precision are negligible with respect to other sources of noise (e.g., label noise). We found that $\|\boldsymbol{w}_{\text{SSGPR}} - \boldsymbol{w}_{\text{I-SSGPR}}\|$ remains below $1 \cdot 10^{-6}$ even after performing 5 million updates on the Cross 2D dataset. Interestingly, both the Cholesky and Sherman–Morrison update formula attained similar accuracy.

[6] This documentation is available at: http://www.ipab.inf.ed.ac.uk/slmc/software/lwpr/lwpr_doc.pdf.

Please cite this article in press as: Gijsberts, A., & Metta, G. Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression. Neural Networks (2012), doi:10.1016/j.neunet.2012.08.011

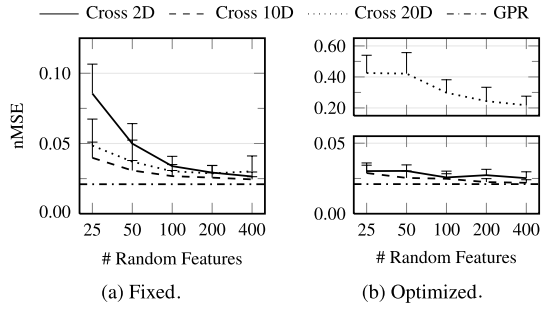

Fig. 2. Test error for SSGPR with respect to the number of sparse spectrum features on three variants of the cross dataset when (a) the frequencies are fixed and (b) the frequencies are optimized together with the other hyperparameters. The reported results are the average over 25 runs, with error bars indicating a unit standard deviation. The test error of GPR is nearly identical on all three datasets, and therefore reported only once.

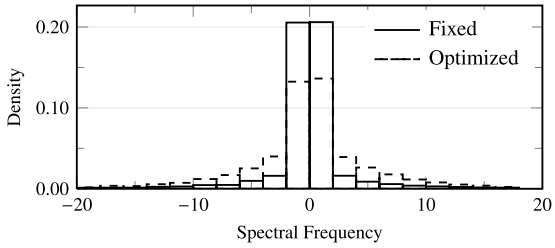

Fig. 3. Distribution of the ℓ-scaled spectral frequencies $\boldsymbol{\Omega}$ for SSGPR^{25} with and without frequency optimization on the Cross 20D dataset. The joint distribution over all 20 input dimensions and over 25 runs is reported.

the number of random features, however, is an adequate measure to alleviate this issue.

The generalization performance of SSGPR in the case of tuned spectral frequencies is impressive on the Cross 2D and Cross 10D datasets, noting that a mere 25 random features are sufficient to achieve competitive results. However, the opposite is true in the case of Cross 20D, for which the performance is significantly inferior with respect to the competing methods. These large test errors in combination with near-zero training errors (not reported) indicate overfitting of the model to the data, as observed as well by Lázaro-Gredilla et al. (2010) on a selected number of datasets. This can be contributed to the difficulty of tuning a large number of hyperparameters in these cases; even for 25 random features the number of hyperparameters is in fact larger than the number of training samples. As seen in Fig. 3, the optimization procedure results in a considerably wider distribution of the (scaled) spectral frequencies. This is another indicator of overfitting, since larger frequencies equate to approximating an ASE kernel with smaller characteristic length-scales (cf. bandwidth).

The results of LWPR can be seen in Fig. 4. Although we were unable to replicate the results reported by Vijayakumar et al. (2005),[7] we clearly see that an increasing number of input dimensions has a detrimental effect on the convergence rate and final performance. LWPR ultimately attains similar performance to GPR and SSGPR on the Cross 2D and Cross 10D datasets, but it requires a large number of iterations over the training samples. Conversely, the two latter methods require only a single pass on the training data. This is a strong indicator of the poor sample complexity of LWPR. Additionally, hyperparameter optimization for LWPR is significantly more involved, requiring "trial-and-error" tuning of multiple hyperparameters for which there is no

[7] This may be due to unreported parameter settings by Vijayakumar et al., or due to the fact that here we consider a single random instance of the dataset.

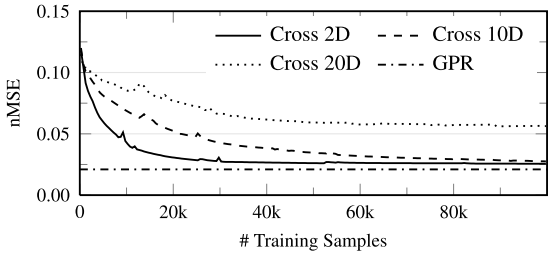

Fig. 4. Convergence of the test error for LWPR with an increasing number of reiterations of the training samples.

intuitive understanding of how they exactly affect generalization performance.

4.2. Robot inverse dynamics

The previous experiment evaluated the learning methods in a batch learning context, in which generalization performance is estimated by means of the error on a separate test set. In most real-life applications, however, the typical i.i.d. assumption is violated and discrepancies between the train and test distribution may result in suboptimal performance of batch models. Incremental learning algorithms, on the other hand, can address these differences in distribution by continuously adapting their model to recent observations. Here we investigate this advantage compared to the batch learning approach at the hand of the real-life problem of modeling inverse dynamics of a robotic manipulator, which relates joint positions $\boldsymbol{q}$, velocities $\dot{\boldsymbol{q}}$, and accelerations $\ddot{\boldsymbol{q}}$ with the corresponding forces $\boldsymbol{F}$ and torques $\boldsymbol{\tau}$ operating on the manipulator. The ability to predict these forces or torques is important for implementing compliant control or to detect external forces exerted on the manipulator. The dynamics can be described analytically using the well-known Rigid Body Dynamics (RBD) formula

$$\boldsymbol{\tau} = \boldsymbol{D}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\,\ddot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) \,,$$

where $\boldsymbol{D}$, $\boldsymbol{C}$ and $\boldsymbol{g}$ are the inertial, Coriolis, and gravity terms, respectively (Spong, Hutchinson, & Vidyasagar, 2006). Unfortunately, this formulation has limited applicability on real-life robots due to the difficulty of accurately determining the various kinematic and dynamic parameters. Moreover, many modern lightweight robots have significant additional non-linear dynamics beyond the RBD model, such as actuator dynamics (e.g., due to gearboxes), routing of cables, and protective covers. Table 2 gives an overview of three large-scale robot dynamics datasets used for our experiments, ranging from several minutes up to over an hour of continuous robot operation. The first dataset is a larger variant of the Sarcos dataset, in which the manipulator performs various rhythmic movement patterns. The second and third datasets have been collected from the upper torso humanoid James (Jamone, Nori, Metta, & Sandini, 2006) and the full body humanoid iCub (Metta et al., 2010), respectively, by moving the manipulator to random Cartesian coordinates in the robot's workspace at constant intervals. In these latter two cases the task is to predict forces and torques as measured in a single proximal F/T sensor positioned just below the shoulder joints, while in the Sarcos dataset the goal is to predict the torque in each individual joint.

The experimental setup resembles the setup for the synthetic datasets as explained in Section 4.1 and I-SSGPR is again compared directly with LWPR and GPR. In order to have a sufficiently large number of samples in the "online" test phase, the traditional subdivision of training and test set of the Sarcos dataset has been reversed. The smaller training set is used exclusively for hyperparameter training and, in the case of GPR, to train a batch

Please cite this article in press as: Gijsberts, A., & Metta, G. Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression. Neural Networks (2012), doi:10.1016/j.neunet.2012.08.011

Table 2
Datasets used for the incremental dynamics experiments and their properties.

Robot	Input	Output	Train	Test	Split type	Freq. (Hz)	Motion	Source
Sarcos	$[q, \dot{q}, \ddot{q}] \times 7$	$\tau \times 7$	4 449	44 484	Interleaved	100	Rhythmic	Vijayakumar and Schaal (2000)
James	$[q, \dot{q}, \ddot{q}] \times 4$	$[F, \tau]_{x,y,z}$	15 000	195 977	Sequential	50	Random	Fumagalli et al. (2010)
iCub	$[q, \dot{q}, \ddot{q}] \times 4$	$[F, \tau]_{x,y,z}$	15 000	72 850	Sequential	50	Random	Gijsberts and Metta (2011)

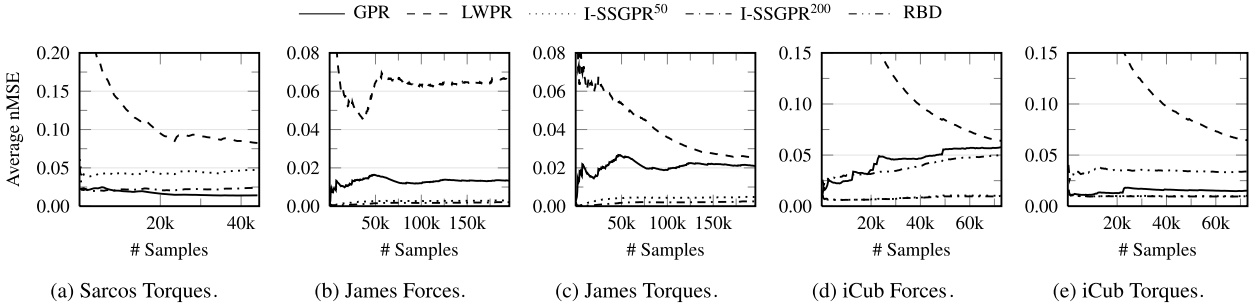

Fig. 5. Convergence of the average one-step-ahead prediction error of the GPR, LWPR, and I-SSGPR methods on the Sarcos, James, and iCub datasets. In case of the latter dataset, the RBD method is included as an additional reference. The standard deviation over the various runs of I-SSGPR is negligible and error bars are therefore omitted for clarity.

model. For the James and iCub datasets, only the first 15 k samples are reserved for this purpose. The reason for this particular split is (1) to have a large number of samples to accurately measure performance in the online phase and (2) to resemble as closely as possible a realistic employment of learning methods in this application domain. Computational requirements of I-SSGPR and GPR are reduced by learning multiple output models using a shared set of hyperparameters. In short, we compute a single kernel or covariance matrix, which is subsequently used to solve multiple weight vectors concurrently (i.e., one for each output) at negligible additional cost. All 7 outputs of the Sarcos dataset are learned using a single model, whereas two models are trained for each of the James and iCub datasets (i.e., one for forces and one for torques). Furthermore, an analytical RBD model computed using the Recursive Newton–Euler Algorithm is included in the experiments for the iCub dataset as an additional reference method (Ivaldi et al., 2011).

4.2.1. Generalization performance

The generalization performance for the experiments is quantified by the online nMSE computed over the one-step-ahead predictions and then averaged over multiple output dimensions. More precisely, at each time step T and given P outputs (i.e., 7 for Sarcos and 3 for each forces and torques for iCub and James) we compute the average error

$$e_T = \frac{1}{P}\frac{1}{T}\sum_{p=1}^{P}\sum_{t=1}^{T}\frac{\left(y_{t,p} - f_{t-1,p}(\boldsymbol{x}_t)\right)^2}{s_p^2},$$

where p indexes the output dimension, s_p^2 denotes the sample variance over the total test set, $f_{t-1,p}$ is the prediction function obtained after the $(t-1)$-th update, and $y_{t,p}$ is the label for the p-th output at the t-th sample corresponding to input $\boldsymbol{x}_t$.

Fig. 5(a) shows how the online nMSE develops on the Sarcos dataset for the two reference methods and I-SSGPR with 50 and 200 sparse spectrum features (henceforth denoted as I-SSGPR^{50} and I-SSGPR^{200}, respectively). These results clearly demonstrate that our method converges almost instantaneously and that generalization performance improves with an increasing number of random features. In fact, I-SSGPR^{200} attains performance that is on a par with GPR. LWPR, on the other hand, is characterized also here by slow convergence and its performance lacks significantly with respect to GPR and both instances of I-SSGPR. Fig. 5(b)–(e) demonstrate similar results for LWPR on the James and iCub datasets as well. For these datasets, however, we may observe different behavior for GPR and I-SSGPR as compared to the Sarcos dataset; I-SSGPR outperforms GPR within 1000 samples and obtains considerably better final performance. This different observation can be attributed to the different sampling strategies. The Sarcos dataset is split in training and test sets by subsampling the total dataset at different intervals and there is consequently a strong correspondence between both subsets. For the James and iCub datasets, on the other hand, the training set consists of the first 15k samples of the total dataset. The presence of temporal drift will cause a divergence in the probability distribution of training and test sets, such that training data is no longer entirely representative for subsequent test data. This is evident from the fact that the average errors of GPR and the RBD model actually increase over time. Incremental methods are much less affected by this divergence, since any test data subsequently becomes training data as well, therefore allowing these methods to gradually adapt to the changes in the distribution.

The analytical RBD model demonstrates acceptable performance with respect to the learned models on the iCub dataset. It consistently outperforms LWPR and also demonstrates better performance than GPR when predicting the force components. An advantage of the analytical model is that it incorporates knowledge of the entire input domain, whereas the learned GPR model is restricted to information present in the training data. I-SSGPR, however, utilizes all available samples for training and shows superior performance over all competing methods. This can be considered a clear indication that a learned model may still be preferred when analytical models are in fact available for a problem.

4.2.2. Computational requirements

Applications that require dynamical models are commonly characterized by (hard) real-time constraints. For example, in the case of James and iCub humanoids, the F/T sensor is sampled at a frequency of around 50 Hz, meaning that predictions taking longer than 20 ms are based on outdated information. These delays would negatively impact control loops that use these predictions. Fig. 6 shows the per-sample timing for the considered methods with respect to the number of online samples on all three datasets. Note that for the incremental methods the time spent for a single sample includes the prediction as well as the subsequent model update.

Please cite this article in press as: Gijsberts, A., & Metta, G. Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression. Neural Networks (2012), doi:10.1016/j.neunet.2012.08.011

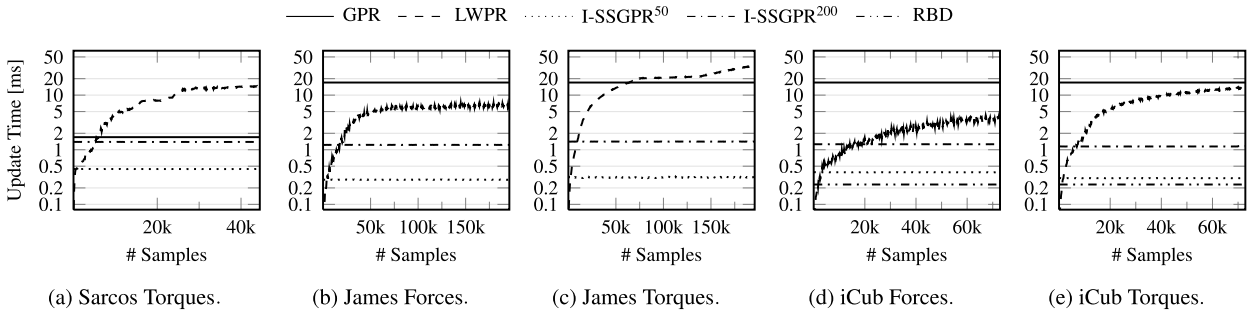

Fig. 6. Sample time with respect to the number of training samples. The results for GPR and RBD only include the prediction time, whereas the timing for LWPR and I-SSGPR also includes the model update. Note that the *y*-axis is in log scale and that data for RBD is only available for the iCub dataset.

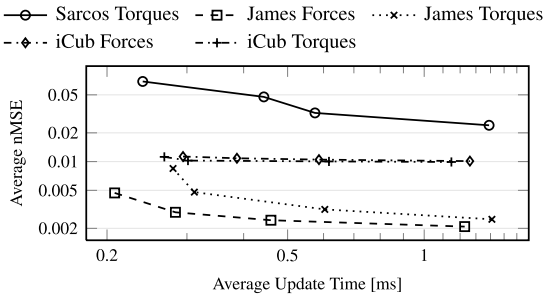

Fig. 7. Final average nMSE of I-SSGPR with (from left to right) $D \in \{25, 50, 100, 200\}$ versus the computational requirements per update on the Sarcos, James, and iCub datasets. Note the log scale on both axes.

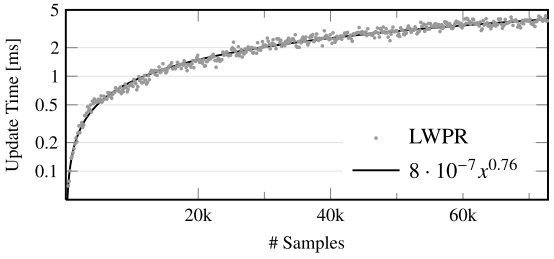

Fig. 8. Per-sample update time of LWPR with respect to the number of samples for the force components of the iCub dataset. The solid line indicates the empirical estimate of the complexity obtained by fitting an exponential curve to the measurements. Note the log scale on the *y*-axis.

As described in Section 3.4, the timing of I-SSGPR remains perfectly constant over time, making this method suitable for use in real-time control loops. Furthermore, the time spent on each sample ranges from approximately 300 μs to 1.5 ms for 50 to 200 sparse spectrum features, respectively, and is thus well within the target limit of 20 ms. In fact, I-SSGPR^{50} is only marginally slower than the RBD model on the iCub dataset.[8] Given the superior predictive accuracy of I-SSGPR^{50}, there is little reason to prefer the analytical RBD model over an autonomously learned I-SSGPR model.

The effect of the number of features D on both performance and computational cost is made explicit in Fig. 7. We can observe that generalization performance improves monotonically with D until reaching an asymptote, where the rate of convergence depends to some extent on the dataset. For instance, increasing D does not result in noteworthy performance gains on the iCub dataset. Regardless, this observed behavior confirms that the number of random features D effectively balances the trade-off between accuracy and computational cost. For typical applications, this parameter can therefore be set to the maximum possible value given the timing constraints.

Although efficient with a small number of samples, the per-sample timings of LWPR are characterized by a steep increase. Not only do these exceed 20 ms in some cases, their unpredictability make LWPR unsuitable for real-time use. For instance, the prediction times seem to have stabilized after approximately 80k samples in Fig. 6(c), only to start increasing again around 150k samples. In Fig. 8, we observe an $\mathcal{O}\left(x^{0.76}\right)$ scaling behavior of LWPR on the entire iCub dataset as calculated using an empirical estimate (Goldsmith, Aiken, & Wilkerson, 2007). The timing behavior of LWPR could be improved by tuning the thresholds for creation and removal of receptive fields, or by considering only the k random fields nearest to the previous "winner" (e.g., Vijayakumar et al., 2005). In an extreme case, for instance, the model could guarantee $\mathcal{O}(1)$ scaling by limiting the maximum number of receptive fields. However, this tuning is cumbersome and will likely have a negative effect on generalization performance. Moreover, the exact balance between creation and removal of receptive fields is data-dependent and further assumptions on the data may be necessary to make formal guarantees. I-SSGPR, on the other hand, has $\mathcal{O}(1)$ scaling "by design" and scales as $\mathcal{O}\left(D^2 + Dn\right)$ with respect to the numbers of inputs n and features D. Its computational requirements are therefore predictable and constant for any given parameter configuration and data distribution.

4.3. Visuo-motor coordination

The last experiment concerns learning the association between head and arm posture, and the corresponding visual response of the humanoid's own hand. This particular sensorimotor mapping is useful for a number of robotic tasks, such as gaze control or visually guided reaching and grasping (see Natale, Nori, Sandini, & Metta, 2007, and references therein). The forward model of this association can be defined as $g : \{\boldsymbol{q}_{\text{arm}}, \boldsymbol{q}_{\text{head}}\} \mapsto \boldsymbol{\imath}$, where $\boldsymbol{q}_{\text{arm}} \in \mathbb{R}^7$ and $\boldsymbol{q}_{\text{head}} \in \mathbb{R}^6$ contain joint angles for respectively the 7-DoF arm and the head configuration (i.e., pitch, roll, yaw, common tilt, common version, and vergence). Furthermore, the image representation $\boldsymbol{\imath} = \left\{u_{\text{left}}, v_{\text{left}}, u_{\text{right}}, v_{\text{right}}\right\} \in \mathbb{N}_+^4$ contains the (approximate) horizontal and vertical coordinates of the center of the hand in the left and right image planes. Obtaining the inverse model g^{-1} is typically more interesting from a practical perspective, since it computes the action required to drive the robot to a desired image representation $\boldsymbol{\imath}$. Unfortunately, constructing the inverse mapping g^{-1} is an ill-posed learning problem due to redundant DoF in advanced humanoid robots. We therefore restrict our attention to the forward learning problem.

[8] This comparison is not entirely fair: the RBD model is implemented entirely in C++ while the I-SSGPR model is partially implemented in Python.

Please cite this article in press as: Gijsberts, A., & Metta, G. Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression. Neural Networks (2012), doi:10.1016/j.neunet.2012.08.011

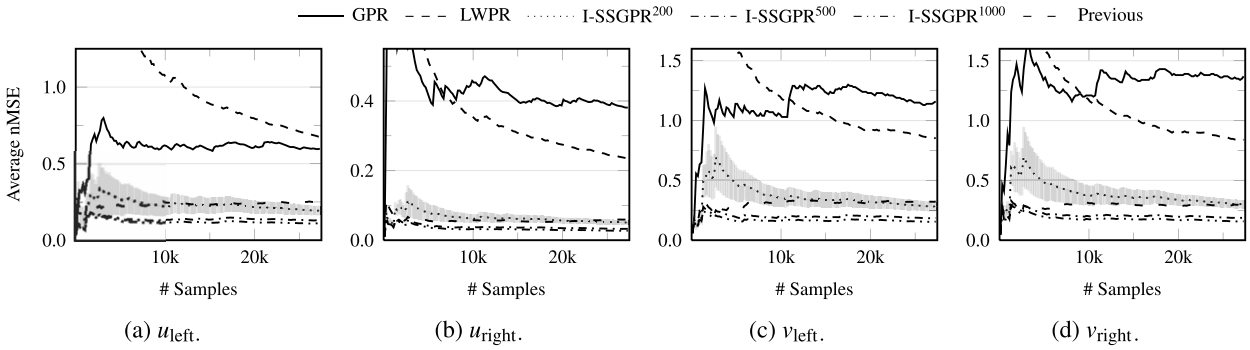

Fig. 9. Convergence of the average one-step-ahead prediction error of the considered methods when predicting the visual location of the hand in terms of u and v coordinates in the left and right image planes. The shared area indicates a unit standard deviation for I-SSGPR^{200}, while the standard deviation is negligible and therefore omitted for I-SSGPR^{500} and I-SSGPR^{1000}.

A total of 429 acquisition epochs were collected from the iCub humanoid consisting of 37 476 samples, where the first 10k samples are used for training and the remainder for testing. In each epoch, the hand is first moved to a random Cartesian position, after which the head and eyes are visually controlled to center the hand in both left and right image planes. It follows that the sequence of samples is weakly dependent, since samples are strongly dependent on a small time-scale (i.e., within epochs), while they become independent on a larger time-scale (i.e., among epochs). The trivial function $f(\boldsymbol{x}_t) = \boldsymbol{y}_{t-1}$, which exploits this dependence by predicting the revealed output of the previous sample, is therefore used as an additional reference benchmark. Furthermore, imperfect localization of the hand in the image frames causes a relatively high level of noise, and in some occasions erroneous output labels. These complications are interesting from a machine learning perspective, since it allows evaluation of learning methods under particularly difficult conditions.

The average online nMSE for the four output coordinates in Fig. 9 confirm the inherent difficulty for learning methods on this dataset. In particular, we observe that GPR performs poorly in all cases with a final nMSE that ranges approximately from 0.4 to 1.5. This leads to the conclusion that the 10k training samples contain highly redundant information and are not sufficient to construct a satisfactory model. Although LWPR outperforms GPR for three out of four outputs, its performance is still significantly worse than the reference method that simply predicts the previous outputs. This inability to compete with a trivial predictor leads to the conclusion that LWPR cannot be used on this dataset. The results are more positive for I-SSGPR, which demonstrates stable and superior performance for all outputs. In the most "economic" configuration of 200 sparse spectrum features, its performance is roughly similar to the reference method. Increasing the number of features to $D = 500$ or $D = 1000$, however, causes a significant improvement in learning performance and results in an overall nMSE between 0.02 and 0.16. In these cases, I-SSGPR unequivocally outperforms the reference method, demonstrating that it can be used successfully in this challenging setting of highly dependent and noisy data.

5. Conclusions

The design objectives of I-SSGPR were (1) a theoretically supported foundation, (2) computational efficiency and constant update complexity, and (3) practical convenience. Rather than developing a novel algorithm from the ground up, our method is based on the thoroughly studied GPR algorithm, therefore ensuring a solid theoretical foundation. The typical linear dependency of standard GPRs on the number of training samples is avoided by explicitly performing a finite-dimensional feature mapping that approximates the kernel function. Consequently, efficient and exact incremental update routines can be used and, contrary to related work, no additional mechanisms are required to constrain computational requirements. This bounded update complexity facilitates open-ended learning as well as use in a (hard) real-time setting. Employment in resource constrained environments, such as embedded systems, is further facilitated by the algorithmic simplicity of the approach. As a consequence, the methods are easily understood, implemented, and deployed in practice. Other features that contribute to practical convenience are a limited number of hyperparameters and a principled methodology (i.e., likelihood optimization) to optimize these hyperparameters automatically. Furthermore, a single hyperparameter controls the dimensionality of the feature mapping and therefore the tradeoff between computational cost and predictive accuracy. Generalization performance can therefore be maximized easily while conforming to domain specific timing constraints.

A number of synthetic and real robot dynamics datasets were used to compare the empirical performance of I-SSGPR with LWPR and batch GPR. These experiments demonstrate that our method significantly outperforms LWPR in terms of predictive accuracy as well as computational requirements. In addition, it outperforms an analytical RBD model by several factors in terms of predictive accuracy, while having comparable computational requirements (less than 1 ms per update). This confirms the computational efficiency of the proposed method. Furthermore, batch GPR was found to be inferior to I-SSGPR when used in a realistic learning setting, in which an initial subset of the samples was used for training and the remaining samples were used for testing.

There are a number of interesting directions for future work. Within the robotics domain, it would be interesting to further investigate the use of these methods in the context of model-based reinforcement learning. The availability of a predictive variance in I-SSGPR could potentially be used to guide the tradeoff between exploration versus exploitation, as demonstrated previously by Strehl and Littman (2008) using related learning methods. Moreover, although we have emphasized robotics applications in this work, it would also be interesting to investigate the performance of our method in other application domains that are incremental by nature, such as time series modeling or adaptive filtering.

Acknowledgments

This work was supported by European Commission grants ITALK (FP7-ICT-214668) and XPERIENCE (FP7-ICT-270273). The authors would like to thank Alberto Parmiggiani, Matteo Fumagalli, Marco Randazzo, and Francesco Nori for making the F/T sensor available on the iCub, while Ugo Pattacini and Carlo Cilliberto are

thanked for their help on collecting the hand-eye coordination dataset. Lastly, we are grateful to Barbara Caputo as well as the anonymous reviewers for providing helpful comments.

References

Björck, Å (1996). *Numerical methods for least squares problems*. Philadelphia: SIAM.
Bochner, S. (1933). Monotone Funktionen, Stieltjessche Integrale und Harmonische Analyse. *Mathematische Annalen*, *108*, 378–410.
Bordes, A., Ertekin, S., Weston, J., & Bottou, L. (2005). Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, *6*, 1579–1619.
Cauwenberghs, G., & Poggio, T. (2001). Incremental and decremental support vector machine learning. In *Advances in neural information processing systems 13* (pp. 409–415). MIT Press.
Cavallanti, G., Cesa-Bianchi, N., & Gentile, C. (2007). Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, *69*, 143–167.
Cesa-Bianchi, N., & Gentile, C. (2008). Improved risk tail bounds for on-line algorithms. *IEEE Transactions on Information Theory*, *54*, 386–390.
Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, learning, and games*. New York, NY, USA: Cambridge University Press.
Crammer, K., Kandola, J. S., & Singer, Y. (2004). Online classification on a budget. In *Advances in neural information processing systems, vol. 16*. Cambridge, MA, USA: MIT Press.
Csató, L., & Opper, M. (2002). Sparse on-line Gaussian processes. *Neural Computation*, *14*, 641–668.
Dekel, O., Shalev-Shwartz, S., & Singer, Y. (2008). The Forgetron: a kernel-based perceptron on a budget. *SIAM Journal on Computing*, *37*, 1342–1372.
Dongarra, J. J., Bunch, J. R., Moler, C. B., & Stewart, G. W. (1979). *LINPACK users' guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics.
Droniou, A., Ivaldi, S., Stalph, P., Butz, M., & Sigaud, O. (2012). Learning velocity kinematics: experimental comparison of on-line regression algorithms. In *Proceedings robotica* (pp. 15–20).
Engel, Y., Mannor, S., & Meir, R. (2002). Sparse online greedy support vector regression. In *Proceedings of the 13th European conference on machine learning*, ECML '02. (pp. 84–96). London, UK: Springer-Verlag.
Engel, Y., Mannor, S., & Meir, R. (2004). The kernel recursive least squares algorithm. *IEEE Transactions on Signal Processing*, *52*, 2275–2285.
Fumagalli, M., Gijsberts, A., Ivaldi, S., Jamone, L., Metta, G., Natale, L., et al. (2010). Learning to exploit proximal force sensing: a comparison approach. In *From motor learning to interaction learning in robots* (pp. 149–167). Springer.
Gijsberts, A., & Metta, G. (2011). Incremental learning of robot dynamics using random features. In *IEEE International conference on robotics and automation* (pp. 951–956).
Goldsmith, S. F., Aiken, A. S., & Wilkerson, D. S. (2007). Measuring empirical computational complexity. In *Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering* (pp. 395–404). New York, NY, USA: ACM.
Golub, G. H., & Loan, C. F. V. (1996). *Matrix computations* (3rd ed.). Baltimore, MD, USA: Johns Hopkins University Press.
Ivaldi, S., Fumagalli, M., Randazzo, M., Nori, F., Metta, G., & Sandini, G. (2011). Computing robot internal/external wrenches by means of inertial, tactile and f/t sensors: theory and implementation on the iCub. In *Proceedings of the 11th IEEE-RAS international conference on humanoid robots*.
Jamone, L., Nori, F., Metta, G., & Sandini, G. (2006). James: a humanoid robot acting over an unstructured world. In *International conference on humanoid robots* (pp. 143–150).
Kersting, K., Plagemann, C., Pfaff, P., & Burgard, W. (2007). Most likely heteroscedastic Gaussian process regression. In *Proceedings of the 24th international conference on machine learning*, ICML '07. (pp. 393–400). New York, NY, USA: ACM.
Kivinen, J., Smola, A. J., & Williamson, R. C. (2004). Online learning with kernels. *IEEE Transactions on Signal Processing*, *52*, 2165–2176.
Klanke, S., Vijayakumar, S., & Schaal, S. (2008). A library for locally weighted projection regression. *Journal of Machine Learning Research*, *9*, 623–626.
Lázaro-Gredilla, M., Quiñonero-Candela, J., & Figueiras-Vidal, A.R. (2007). Sparse spectral sampling Gaussian processes. Technical Report MSR-TR-2007-152. Microsoft Research.
Lázaro-Gredilla, M., Quiñonero-Candela, J., Rasmussen, C. E., & Figueiras-Vidal, A. R. (2010). Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research*, *11*, 1865–1881.
Ma, J., Theiler, J., & Perkins, S. (2003). Accurate on-line support vector regression. *Neural Computation*, *15*, 2683–2703.
MacKay, D. J. C. (1995). Probable networks and plausible predictions — a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, *6*, 469–505.
Martin, M. (2002). On-line support vector machine regression. In *Proceedings of the 13th European conference on machine learning*, ECML '02. (pp. 282–294). London, UK: Springer-Verlag.
Metta, G., Natale, L., Nori, F., Sandini, G., Vernon, D., Fadiga, L., et al. (2010). The iCub humanoid robot: an open-systems platform for research in cognitive development. *Neural Networks*, *23*, 1125–1134.
Natale, L., Nori, F., Sandini, G., & Metta, G. (2007). Learning precise 3D reaching in a humanoid robot. In *IEEE International conference of development and learning* (pp. 324–329).
Nguyen-Tuong, D., Seeger, M. W., & Peters, J. (2009). Model learning with local Gaussian process regression. *Advanced Robotics*, *23*, 2015–2034.
Orabona, F., Keshet, J., & Caputo, B. (2009). Bounded kernel-based online learning. *Journal of Machine Learning Research*, *10*, 2643–2666.
Rahimi, A., & Recht, B. (2008a). Random features for large-scale kernel machines. In *Advances in neural information processing systems 20* (pp. 1177–1184). Cambridge, MA: MIT Press.
Rahimi, A., & Recht, B. (2008b). Uniform approximation of functions with random bases. In *Allerton conference on communication control and computing*, Allerton08 (pp. 555–561).
Rasmussen, C. E., & Williams, C. K. I. (2005). Gaussian processes for machine learning. In *Adaptive computation and machine learning*, The MIT Press.
Sayed, A. H. (2008). *Adaptive filters*. Wiley-IEEE Press.
Schaal, S., & Atkeson, C. G. (1998). Constructive incremental learning from only local information. *Neural Computation*, *10*, 2047–2084.
Schneider, M., & Ertel, W. (2010). Robot learning by demonstration with local Gaussian process regression. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 255–260).
Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge University Press.
Sigaud, O., Salaün, C., & Padois, V. (2011). On-line regression algorithms for learning mechanical models of robots: a survey. *Robotics and Autonomous Systems*, *59*, 1115–1129.
Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2006). *Robot modeling and control*. New York, USA: Wiley.
Strehl, A., & Littman, M. (2008). Online linear regression and its application to model-based reinforcement learning. In *Advances in neural information processing systems 20* (pp. 1417–1424). Cambridge, MA: MIT Press.
Vijayakumar, S., D'souza, A., & Schaal, S. (2005). Incremental online learning in high dimensions. *Neural Computation*, *17*, 2602–2634.
Vijayakumar, S., & Schaal, S. (2000). Locally weighted projection regression: an $O(n)$ algorithm for incremental real time learning in high dimensional spaces. In *Proceedings of the 17th international conference on machine learning*, ICML '00 (pp. 288–293).
Vijayakumar, S., & Wu, S. (1999). Sequential support vector classifiers and regression. In *Proceedings of the third ICSC symposia on intelligent industrial automation and soft computing*, IIA'99, SOCO'99 (pp. 610–619). ICSC Academic Press.
Vovk, V., Gammerman, A., & Shafer, G. (2005). *Algorithmic learning in a random world*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.
Weston, J., Bordes, A., & Bottou, L. (2005). Online (and offline) on an even tighter budget. In *Proceedings of the tenth international workshop on artificial intelligence and statistics* (pp. 413–420). Society for Artificial Intelligence and Statistics.
Zhdanov, F., & Kalnishkan, Y. (2010). An identity for kernel ridge regression. In *Proceedings of the 21st international conference on algorithmic learning theory*, ALT'10 (pp. 405–419). Berlin, Heidelberg: Springer-Verlag.

Please cite this article in press as: Gijsberts, A., & Metta, G. Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression. Neural Networks (2012), doi:10.1016/j.neunet.2012.08.011