Lifelong Learning with Gaussian Processes

Christopher Clingerman and Eric Eaton(✉)

Department of Computer and Information Science,
University of Pennsylvania, Philadelphia, PA 19104, USA
{chcl,eeaton}@seas.upenn.edu

Abstract. Recent developments in lifelong machine learning have demonstrated that it is possible to learn multiple tasks consecutively, transferring knowledge between those tasks to accelerate learning and improve performance. However, these methods are limited to using linear parametric base learners, substantially restricting the predictive power of the resulting models. We present a lifelong learning algorithm that can support non-parametric models, focusing on Gaussian processes. To enable efficient online transfer between Gaussian process models, our approach assumes a factorized formulation of the covariance functions, and incrementally learns a shared sparse basis for the models' parameterizations. We show that this lifelong learning approach is highly computationally efficient, and outperforms existing methods on a variety of data sets.

Keywords: Lifelong machine learning · Online multi-task learning
Gaussian process

1 Introduction

Recent advances in lifelong machine learning [3,23] have shown that it is now possible to learn tasks consecutively and obtain equivalent model performance to batch multi-task learning (MTL) [14,15] with dramatic computational speedups. Lifelong learning methods have been developed for either classification and regression [23] or reinforcement learning [3,6] domains, including support for autonomous cross-domain mapping of the learned knowledge [5]. These multi-task and lifelong learning methods maintain a repository of learned knowledge that acts as a basis over the model space, and is shared between the task models to facilitate transfer between tasks. Although effective, these methods are currently limited to using linear parametric base learners, substantially limiting the predictive power of the resulting models.

We present the first lifelong learning algorithm that supports non-parametric models through Gaussian process (GP) regression. GPs have been used successfully in MTL [1,2,4,21,28,29], but lifelong learning with GPs has not yet been explored. To enable transfer between the GP task models, we assume a factorized formulation of the models' parameterizations using a shared sparse basis, and incrementally learn a repository of shared knowledge that acts as that basis to underly the covariance functions. This shared knowledge base is updated with each new task, serving both to accelerate learning of GP models for future tasks via knowledge transfer and to refine the models of known tasks. This process is computationally efficient, and we provide theoretical analysis of the convergence of our approach. We demonstrate the effectiveness of GP lifelong learning on a variety of data sets, outperforming existing GP MTL methods and lifelong learning with linear models.

2 Related Work

Gaussian processes (GP) have proven to be effective tools for modeling spatial, time-varying, or nonlinear data [22], providing substantially more predictive power than linear models. However, the model complexity of a standard GP places restrictions on the amount of data that can be processed in a batch multi-task setting. Typical (albeit naïve) computation requires $O(n^3)$ time, where n is the number of training data instances. In single-task settings, many interesting approaches such as input sparsification [24], hierarchical GPs with input clustering [18], and distributed GPs for large-scale regression [9] have been investigated for reducing computation time while maintaining prediction accuracy. Other work has focused on scaling the amount of data that GPs can reasonably handle [7,20]. While these single-task GP learning methods do not consider MTL or lifelong learning scenarios, their approaches could be used as the base learners in our framework to (1) further scale the amount of data that can be handled for individual tasks, and (2) reduce the data storage requirements for previous tasks.

Several works have tackled the MTL setting with GP predictors [1,2,4,21,28,29]. These methods attempt to learn some form of shared knowledge among related tasks in a batch setting. However, the manner in which these models form connections among tasks increases complexity and, incidentally, computation time. Consequently, these methods are inappropriate for the lifelong learning setting, in which tasks arrive consecutively and the models must be repeatedly updated. Our approach, in contrast, considers the case where tasks are observed online and utilizes a factorized model that is more computationally efficient. Also, in direct comparison with batch MTL using GPs [4], we show that our method requires significantly less computation time while maintaining comparable accuracy.

Lifelong machine learning [8] has similarly seen much interest and development over the past several years in the aforementioned settings of classification, regression, and reinforcement learning [3,6,10,19,23,27], with applications to robotics [12,25], user modeling [13], and learning of structured information [16]. A popular choice for the foundation of these algorithms is linear parametric models. While linear models are simple and computationally efficient, they lack the predictive power of GPs. As a natural yet non-trivial extension of this prior work, we have devised a novel approach to lifelong learning using GPs as the base learning algorithm to merge, in some sense, the best of both worlds—our approach combines the predictive power of GPs in the multi-task setting while utilizing the computational efficiency and longevity of lifelong learning.

3 Background on Gaussian Processes

A Gaussian process (GP) is a distribution over functions $g(\mathbf{x})$, where the distribution is determined solely by mean and covariance functions [22]:

$$g(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), \mathbf{K}) \ , \tag{1}$$

with mean $m(\cdot)$ and covariance matrix $\mathbf{K}$. GPs can be used for modeling labeled data by assuming that the data are samples drawn from a multivariate Gaussian distribution. Given a set of labeled training instances $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ with $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$ and $y_i \in \mathbb{R}$, GP regression models the likelihood as a Gaussian with $\mathrm{P}(\mathbf{y} \mid \mathbf{X}, g) = \mathcal{N}(g, \sigma^2\mathbf{I})$, where $g : \mathcal{X} \mapsto \mathbb{R}$ maps from instances to their corresponding labels. The prior on g is then defined as $\mathrm{P}(g(\mathbf{x}_i) \mid \boldsymbol{\theta}) = \mathcal{GP}(m(\mathbf{x}_i), \mathbf{K})$, where $m(\mathbf{x}_i)$ is the mean function on the input datum $\mathbf{x}_i$, $\mathbf{K}$ is a covariance matrix $[\mathbf{K}]_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ between all pairs of data instances $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$ based on a chosen covariance function, and $\boldsymbol{\theta}$ is a set of parameters of the chosen covariance function. Given a new input datum $\mathbf{x}_*$, the GP predicts the distribution of the corresponding label value y_* as $\mathrm{P}(y_* \mid \mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(\mu_*, \sigma_*)$, where

$$\begin{aligned}\mu_* &= m(\mathbf{x}_*) + \mathbf{k}_*^{\mathsf{T}} \left[\mathbf{K} + \sigma^2\mathbf{I}\right]^{-1} \left(\mathbf{y} - m(\mathbf{x}_*)\right) \ , \\ \sigma_* &= \kappa(\mathbf{x}_*, \mathbf{x}_*) + \sigma^2 - \mathbf{k}_*^{\mathsf{T}} \left[\mathbf{K} + \sigma^2\mathbf{I}\right]^{-1} \mathbf{k}_* \ ,\end{aligned}$$

and $\mathbf{k}_*$ is a vector of covariance values $[\mathbf{k}_*]_i = \kappa(\mathbf{x}_*, \mathbf{x}_i)$ for all $\mathbf{x}_i \in \mathbf{X}$. In other words, GP regression models:

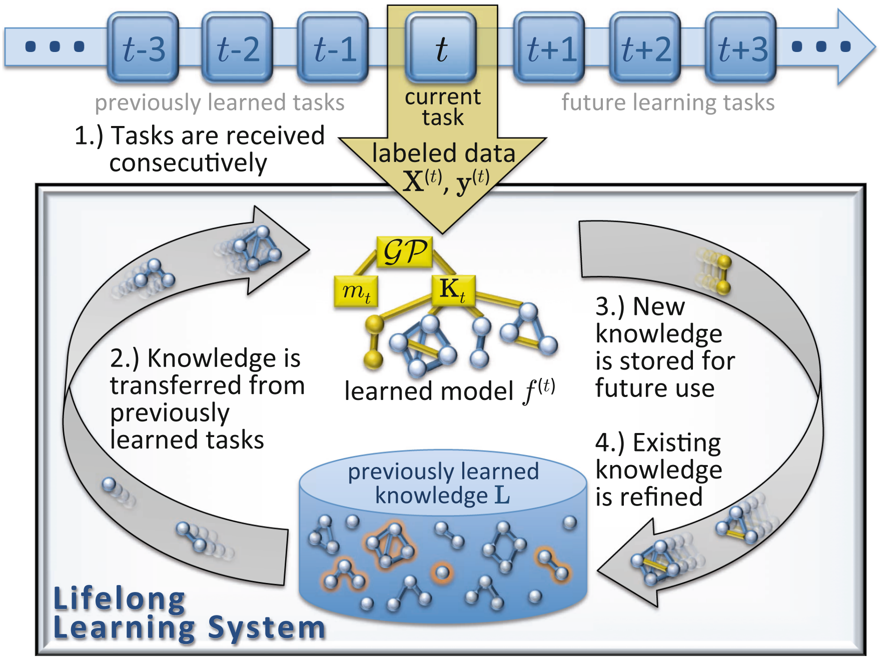
$$\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\mathbf{X}) \\ m(\mathbf{x}_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \mathbf{k}_*^{\mathsf{T}} \\ \mathbf{k}_* & k_{**} \end{bmatrix} \right) \ . \tag{2}$$

4 Lifelong Learning with Gaussian Processes

After first summarizing the lifelong learning problem, we extend GP regression to the multi-task setting and derive our approach for lifelong GP learning. To ensure consistency with the existing literature, we adopt the notational conventions of Ruvolo and Eaton [23].

4.1 The Lifelong Learning Problem

The lifelong learning agent (Fig. 1) faces a series of consecutive learning tasks $\mathcal{Z}^{(1)}, \mathcal{Z}^{(2)}, \ldots, \mathcal{Z}^{(T_{\max})}$. In our setting, each task is a supervised or semi-supervised learning problem $\mathcal{Z}^{(t)} = \left(id^{(t)}, \hat{f}^{(t)}, \mathbf{X}^{(t)}, \mathbf{y}^{(t)}\right)$, where $id^{(t)}$ is a unique task identifier, $\hat{f}^{(t)} : \mathcal{X}^{(t)} \mapsto \mathcal{Y}^{(t)}$ defines an unknown mapping from an instance space $\mathcal{X}^{(t)} \subseteq \mathbb{R}^d$ to the label space $\mathcal{Y}^{(t)}$. Typically, $\mathcal{Y}^{(t)} = \{-1, +1\}$ for classification tasks and $\mathcal{Y}^{(t)} = \mathbb{R}$ for regression tasks. Each task t has n_t training instances $\mathbf{X}^{(t)} \in \mathbb{R}^{d \times n_t}$ with corresponding labels $\mathbf{y}^{(t)} \in {\mathcal{Y}^{(t)}}^{n_t}$ given by $\hat{f}^{(t)}$. *A priori*, the lifelong learner does not know the total number of tasks $T_{\max}$, the task distribution, or their order.



t-3
t-2
t-1
t
t+1
t+2
t+3
previously learned tasks
current task
future learning tasks
1.) Tasks are received consecutively
labeled data $\mathbf{X}^{(t)}, \mathbf{y}^{(t)}$
$\mathcal{GP}$
m_t
$\mathbf{K}_t$
learned model $f^{(t)}$
2.) Knowledge is transferred from previously learned tasks
3.) New knowledge is stored for future use
4.) Existing knowledge is refined
previously learned knowledge $\mathbf{L}$
Lifelong Learning System

Fig. 1. The lifelong learning process with GPs (adapted from [23]).

Each time step, the agent receives a batch of labeled training data for some task t, either a new task or as additional data for a previously learned task. Let T denote the number of unique tasks the agent has encountered so far $(0 \leq T \leq T_{\max})$. At any time, the agent may be asked to make predictions on instances from any previously learned task. Its goal is to construct task models $f^{(1)}, \ldots, f^{(T)}$ where each $f^{(t)} : \mathbb{R}^d \mapsto \mathcal{Y}^{(t)}$ will approximate $\hat{f}^{(t)}$. Each $f^{(t)}$ must be able to be rapidly updated as the agent encounters additional training data for known tasks, and new $f^{(t)}$'s must be added efficiently as the agent encounters new tasks. We assume that the total number of tasks $T_{\max}$ will be large, and so the algorithm must have a computational complexity to update the task models that scales favorably to numerous tasks.

4.2 Gaussian Process Regression for Multi-task Learning

To extend the GP framework (Sect. 3) to the multi-task and lifelong learning setting, we assume that each task t corresponds to training an individual GP model $f^{(t)}$ from training data $\mathbf{D}^{(t)} = \{(\mathbf{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^{n_t}$ with its own mean $m_t(\cdot)$ and covariance function $\mathbf{K}_t$. Given T tasks, the goal is to learn each task model $f^{(t)}$ such that

$$\mathrm{P}(\mathbf{y}^{(t)} \mid \mathbf{X}^{(t)}, g^{(t)}) \sim \mathcal{N}(g^{(t)}, \sigma_t^2 \mathbf{I})$$
$$g^{(t)} \sim \mathcal{GP}(m_t, \mathbf{K}_t) \quad \forall t \in \{1, \ldots, T\} \ .$$

To share knowledge between task models, we assume a factorized form of the covariance kernels, such that the covariance kernel parameters $\boldsymbol{\theta}^{(t)} \in \mathbb{R}^d$ can be represented as $\boldsymbol{\theta}^{(t)} = \mathbf{L}\mathbf{s}^{(t)}$. The matrix $\mathbf{L} \in \mathbb{R}^{d \times k}$ is shared between all tasks and forms a basis over the space of covariance kernels, capturing reusable chunks of knowledge. Individual tasks' covariance kernels are then defined in this shared basis via the sparse coefficient vectors $\{\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(T)}\}$, with $\mathbf{s}^{(t)} \in \mathbb{R}^k$. This factorized sparse representation has shown success for transfer between linear parameterized models in previous MTL and lifelong learning methods [3,14,15,23], and here we adapt it to the non-parametric GP setting. In this manner, our approach is similar to these other methods in that we employ a parametric formulation in which the prediction function $f^{(t)}(\mathbf{x}) = f(\mathbf{x}; \boldsymbol{\theta}^{(t)})$ for each task t is determined in part by the covariance parameter vector $\boldsymbol{\theta}^{(t)} \in \mathbb{R}^d$, but the model itself is non-parametric. This factorized sparse representation is also somewhat related to techniques for fast GP training using factorized covariance matrices [11,21], but with a different factorization designed to share latent knowledge between task models.

We assume that the tasks are drawn i.i.d., allowing us to define the following MTL objective function for GPs:

$$\min_{\mathbf{L}} \frac{1}{T} \sum_{t=1}^{T} \min_{\mathbf{s}^{(t)}} \left\{ \mathcal{L}\left(f\left(\mathbf{X}^{(t)}; \mathbf{L}\mathbf{s}^{(t)}\right), \mathbf{y}^{(t)}\right) + \mu \|\mathbf{s}^{(t)}\|_1 \right\} + \lambda \|\mathbf{L}\|_{\mathsf{F}}^2 \ , \tag{3}$$

where $(\mathbf{x}_i^{(t)}, y_i^{(t)})$ is the ith labeled training instance for task t, the L_1 norm $\|\cdot\|_1$ is used as a convex approximation to the true vector sparsity of the $\mathbf{s}^{(t)}$ vectors, $\|\cdot\|_{\mathsf{F}}$ is the Frobenius norm to control the complexity of $\mathbf{L}$, and μ and λ are regularization coefficients. The loss function $\mathcal{L}$ optimizes the fit of each task model to the training data. For GP models, we define this loss function as the negative log-marginal likelihood of a standard Gaussian process, which must be minimized to fit the GP:

$$\begin{aligned}\mathcal{L}\Big(f\Big(\mathbf{X}^{(t)}; \mathbf{L}\mathbf{s}^{(t)}\Big), \mathbf{y}^{(t)}\Big) \ &= \ \mathbf{y}^{(t)\mathsf{T}} \Big[\mathbf{K}_t(\mathbf{L}\mathbf{s}^{(t)}) + \sigma_t^2 \mathbf{I}\Big]^{-1} \mathbf{y}^{(t)} \\ &\quad + \log \left|\mathbf{K}_t(\mathbf{L}\mathbf{s}^{(t)}) + \sigma_t^2 \mathbf{I}\right| + n_t \log(2\pi) \ .\end{aligned} \tag{4}$$

Note that Eq. 3 is not jointly convex in $\mathbf{L}$ and the $\mathbf{s}^{(t)}$ vectors, so most MTL methods [14,15] solve related forms of this objective using alternating optimization, repeatedly solving for $\mathbf{L}$ while holding $\mathbf{s}^{(t)}$'s fixed and then optimizing the $\mathbf{s}^{(t)}$'s while holding $\mathbf{L}$ fixed. While effective in determining a locally optimal solution, these MTL approaches are computationally expensive and would require re-optimization as tasks were added incrementally, making them unsuitable for the lifelong learning setting.

Our approach to optimizing Eq. 3 is based upon the Efficient Lifelong Learning Algorithm (ELLA) [23], which provides a computationally efficient method for learning the $\mathbf{s}^{(t)}$'s and $\mathbf{L}$ online over multiple consecutive tasks. Although ELLA can support a variety of parametric linear models, it cannot natively support non-parametric models, limiting its predictive power. In the next section, we develop a lifelong learning approach for the (non-parametric) GP framework, and show that the resulting algorithm provides an efficient method for learning consecutive GP task models.

4.3 Efficient Updates for Lifelong GP Learning

To solve Eq. 3 efficiently in a lifelong learning setting, we first eliminate the inner recomputation of the loss function by approximating it via a sparse-coded solution, following Ruvolo and Eaton [23]. We approximate the loss function via a second-order Taylor expansion around $\boldsymbol{\theta} = \boldsymbol{\theta}^{(t)}$ of $\mathcal{L}\left(f\left(\mathbf{X}^{(t)}; \mathbf{L}\mathbf{s}^{(t)}\right), \mathbf{y}^{(t)}\right)$, where $\boldsymbol{\theta}^{(t)} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}\left(f\left(\mathbf{X}^{(t)}; \boldsymbol{\theta}, \mathbf{y}^{(t)}\right)\right)$. Substituting this expansion into Eq. 3, we obtain

$$\min_{\mathbf{L}} \frac{1}{T} \sum_{t=1}^{T} \min_{\mathbf{s}^{(t)}} \left\{ \|\boldsymbol{\theta}^{(t)} - \mathbf{L}\mathbf{s}^{(t)}\|_{\mathbf{H}^{(t)}}^2 + \mu \|\mathbf{s}^{(t)}\|_1 \right\} + \lambda \|\mathbf{L}\|_{\mathsf{F}}^2 \ , \tag{5}$$

where $\mathbf{H}^{(t)}$ is the Hessian matrix given by

$$\mathbf{H}^{(t)} = \frac{1}{2} \nabla_{\boldsymbol{\theta},\boldsymbol{\theta}}^2 \mathcal{L}\left(f\left(\mathbf{X}^{(t)}; \boldsymbol{\theta}\right), \mathbf{y}^{(t)}\right) \Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} \ , \tag{6}$$

and $\|\mathbf{v}\|_{\mathbf{A}}^2 = \mathbf{v}^\top \mathbf{A} \mathbf{v}$.

The second inefficiency in Eq. 3 involves the need to recompute the $\mathbf{s}^{(t)}$'s whenever we evaluate a new $\mathbf{L}$. This dependency can be simplified by only recomputing the $\mathbf{s}^{(t)}$ for the current task t, leaving all other coefficient vectors fixed. Essentially we have removed the minimization over all $\mathbf{s}^{(t)}$'s in place of a minimization over only the current task's $\mathbf{s}^{(t)}$. Later, we provide convergence guarantees that show that this choice to update $\mathbf{s}^{(t)}$ only when training on task t does not significantly affect the quality of model fit as T grows large. With these simplifications, we can solve the optimization in Eq. 5 incrementally via the following update equations:

$$\mathbf{s}^{(t)} \leftarrow \arg\min_{\mathbf{s}^{(t)}} \ell(\mathbf{L}_m, \mathbf{s}^{(t)}, \boldsymbol{\theta}^{(t)}, \mathbf{H}^{(t)}) \tag{7}$$

$$\mathbf{L}_{m+1} \leftarrow \arg\min_{\mathbf{L}} \hat{g}_m(\mathbf{L}) \tag{8}$$

$$\hat{g}_m(\mathbf{L}) = \lambda \|\mathbf{L}\|_{\mathsf{F}}^2 + \frac{1}{T} \sum_{t=1}^{T} \ell\left(\mathbf{L}, \mathbf{s}^{(t)}, \boldsymbol{\theta}^{(t)}, \mathbf{H}^{(t)}\right) \ , \tag{9}$$

where

$$\ell\left(\mathbf{L}, \mathbf{s}, \boldsymbol{\theta}, \mathbf{H}\right) = \mu \left\|\mathbf{s}\right\|_1 + \|\boldsymbol{\theta} - \mathbf{L}\mathbf{s}\|_{\mathbf{H}}^2 \tag{10}$$

and $\mathbf{L}_m$ corresponds to $\mathbf{L}$ at the algorithm's m-th iteration.

To apply these update equations in practice, given a new task t, we first compute $\boldsymbol{\theta}^{(t)}$ via single-task GP learning on $\mathbf{D}^{(t)}$. Then, we compute the Hessian $\mathbf{H}^{(t)}$, as described in detail in Appendix A. Next, we compute $\mathbf{s}^{(t)}$ and $\mathbf{L}_m$ in a single gradient-descent step. For each iteration of gradient descent, we compute $\mathbf{s}^{(t)}$ using the current $\mathbf{L}_m$ by solving an instance of LASSO. Then, we use $\mathbf{s}^{(t)}$ to find $\nabla\mathbf{L}$ and recompute $\mathbf{L}_m$. This process is repeated until either convergence or a set maximum number of iterations has taken place; typically, this process requires only a few iterations to converge in practice. The equation for $\nabla\mathbf{L}$ is found by deriving Eq. 9 with respect to $\mathbf{L}$, similar to the derivation by Bou Ammar et al. [5], yielding:

$$\nabla\mathbf{L} = \lambda\mathbf{L} + \frac{1}{T}\sum_{t=1}^{T}\left(-\mathbf{H}^{(t)}\boldsymbol{\theta}^{(t)}\mathbf{s}^{(t)\mathsf{T}} + \mathbf{H}^{(t)}\mathbf{L}\mathbf{s}^{(t)}\mathbf{s}^{(t)\mathsf{T}}\right). \tag{11}$$

For computational efficiency, the sum in Eq. 11 is computed incrementally (by updating it for the current task) and stored; it is not necessary to recompute it via summing over all previous tasks. As a final step, we reinitialize any unused columns of $\mathbf{L}_m$. Algorithm 1 presents our complete algorithm for lifelong GP learning, which we refer to as GP-ELLA.

In contrast to linear methods for lifelong learning [3,23], GP-ELLA must explicitly store training data for each individual task to compute the non-parametric kernel values for new instances. To further improve its scalability for lifelong learning with large amounts of data, our approach could also be adapted to employ sparse GPs [24] as the base learners, which store a reduced number of instances per task, or use approximations to the kernel matrix. We started exploring this direction, but found that the adaptation is nontrivial, well beyond the scope of this paper, and so leave it to future work.

4.4 Details of Label Prediction: Recovering $\boldsymbol{\sigma_f^2}$ and Prediction Smoothing

In practice, we minimize the negative log-marginal likelihood in training the task-based GP model over all parameters, including σ_f^2. However, our model of $\boldsymbol{\theta}^{(t)}$ does not include a representation of σ_f^2, so in the label prediction step, we hold out a portion of the training data as a verification test set and perform line search to determine the best possible σ_f^2 parameter given the generated $\boldsymbol{\theta}^{(t)} = \mathbf{L}\mathbf{s}^{(t)}$.

For our label predictions, we also utilize an idea from Nguyen-Tuong et al. [17], where the training data are partitioned into subsets and the final predictions are smoothed using a weighted average of the local model predictions. In our predictions for a task t, we first generate predictions with respect to $\boldsymbol{\theta}^{(t)} = \mathbf{L}\mathbf{s}^{(t)}$. For all other known tasks t', we also generate $\boldsymbol{\theta}^{(t')} = \mathbf{L}\mathbf{s}^{(t')}$ and its corresponding predictions for the current task t. Then, we compute a weighted average of all predictions based on similarities in the $\boldsymbol{\theta}$ vectors, using an exponentially decaying

Algorithm 1. GP-ELLA$(d, k, \lambda, \mu, \gamma)$

```
$T \leftarrow 0, \quad \mathbf{L} \sim \mathcal{N}(0,1)_{d,k}$
while isMoreTrainingDataAvailable() do
    $(\mathbf{X}_{\text{new}}, \mathbf{y}_{\text{new}}, t) \leftarrow$ getNextTrainingData()
    if isNewTask($t$) then
        $T \leftarrow T + 1, \quad \mathbf{X}^{(t)} \leftarrow \mathbf{X}_{\text{new}}, \quad \mathbf{y}^{(t)} \leftarrow \mathbf{y}_{\text{new}}$
    else
        $\mathbf{X}^{(t)} \leftarrow \left[\mathbf{X}^{(t)}\ \mathbf{X}_{\text{new}}\right], \quad \mathbf{y}^{(t)} \leftarrow \left[\mathbf{y}^{(t)}; \mathbf{y}_{\text{new}}\right]$
    end if
    $\left(\boldsymbol{\theta}^{(t)}, \mathbf{H}^{(t)}\right) \leftarrow$ GPLearner$(\mathbf{X}^{(t)}, \mathbf{y}^{(t)})$
    while $\mathbf{s}^{(t)}$ and $\mathbf{L}$ have not converged do
        $\mathbf{s}^{(t)} \leftarrow$ Equation [7]
        $\nabla\mathbf{L} \leftarrow$ Equation [11]
        $\mathbf{L} \leftarrow \mathbf{L} - \gamma\nabla\mathbf{L}$
    end while
    $\mathbf{L} \leftarrow$ reinitializeAllZeroColumns$(\mathbf{L})$
end while
```

L_2-norm weight function. This additional label generation step can be thought of as further smoothing predictions among similar known tasks, as measured by the similarity of their $\boldsymbol{\theta}^{(t)}$'s. In the case where the tasks are different, this additional smoothing step will not detrimentally affect the predictions, since their corresponding $\boldsymbol{\theta}^{(t)}$'s will have low similarity.

4.5 Theoretical Guarantees

This section provides theoretical results that show that GP-ELLA converges and that the simplifications to enable efficient updates have an asymptotically negligible effect on model performance. First, recall that $\hat{g}_T(\boldsymbol{L})$ (as defined by Eq. 9 with $m = T$) represents the cost of $\boldsymbol{L}$ under the current choice of the $\mathbf{s}^{(t)}$'s after GP-ELLA observes T tasks. Let $e_T(\mathbf{L}_T)$ be the MTL objective function as defined by Eq. 3, but for the given specific choice of $\mathbf{L}$ (instead of the optimization over all $\mathbf{L}$). Ruvolo and Eaton [23] showed that:

Proposition 1: The latent basis becomes more stable over time at a rate of $\boldsymbol{L}_{T+1} - \boldsymbol{L}_T = O\left(\frac{1}{T}\right)$.

Proposition 2:

(a) $\hat{g}_T(\mathbf{L}_T)$ converges almost surely (a.s.); and
(b) $\hat{g}_T(\mathbf{L}_T) - e_T(\mathbf{L}_T)$ converges a.s. to 0.

Proposition 1 shows that $\mathbf{L}$ becomes increasingly stable as T increases. Proposition 2 shows that the algorithm converges to a fixed per-task loss on the approximate objective function $\hat{g}_T$, and that the approximate objective function converges to the same value as the MTL objective.

In order for these propositions to apply to GP-ELLA, we must show that it satisfies the following assumptions:

1. The tuples $\left(\mathbf{H}^{(t)}, \boldsymbol{\theta}^{(t)}\right)$ are drawn *i.i.d.* from a distribution with compact support.
2. For all $\boldsymbol{L}$, $\mathbf{H}^{(t)}$, and $\boldsymbol{\theta}^{(t)}$, the smallest eigenvalue of $\boldsymbol{L}_\gamma^\top \mathbf{H}^{(t)} \boldsymbol{L}_\gamma$ is at least κ (with $\kappa > 0$), where γ is the subset of non-zero indices of the vector $\mathbf{s}^{(t)} = \arg\min_{\mathbf{s}} \|\boldsymbol{\theta}^{(t)} - \boldsymbol{L}\boldsymbol{s}\|^2_{\mathbf{H}^{(t)}}$. In this case the non-zero elements of the unique minimizing $\mathbf{s}^{(t)}$ are given by: $\mathbf{s}^{(t)}{}_\gamma = \left(\boldsymbol{L}_\gamma^\top \mathbf{H}^{(t)} \boldsymbol{L}_\gamma\right)^{-1} \left(\boldsymbol{L}_\gamma^\top \mathbf{H}^{(t)} \boldsymbol{\theta}^{(t)} - \mu \boldsymbol{\epsilon}_\gamma\right)$, where the vector $\boldsymbol{\epsilon}_\gamma$ contains the signs of the non-zero entries of $\mathbf{s}^{(t)}$.

To verify the first assumption, we must show that the entries of $\mathbf{H}^{(t)}$ and $\boldsymbol{\theta}^{(t)}$ are bounded with compact support. We can show easily that the Hessian and $\boldsymbol{\theta}^{(t)}$ are contained within a compact region by examining their form, thus verifying the first assumption. The second assumption is a condition upon the sparse coding solution being unique, which holds true under the standard sparse coding assumptions. Therefore, the propositions above apply to GP-ELLA. In particular, since Proposition 2 holds, this verifies that the simplifications made in the optimization process (Sect. 4.3) do not cause GP-ELLA to incur any penalty in terms of the average per-task loss.

Computational Complexity: Each GP-ELLA update requires running a single-task GP on n_t d-dimensional data instances; let $M(d, n_t)$ be the complexity of this operation. Then, the algorithm iteratively optimizes $\boldsymbol{s}^{(t)}$ and $\boldsymbol{L}$ at a cost of $O(k^2 d^3)$ per iteration [23]. This process typically requires very few iterations i, or we can limit the number of iterations i, which works well in practice. Together, this gives GP-ELLA a per-task cost of $O(M(d, n_t) + ik^2 d^3)$.

5 Evaluation

To evaluate GP-ELLA, we analyze its prediction accuracy and computation time in comparison to four alternatives: task-independent GP learners ("Indiv. GPs"), a single GP trained over the entire data set ("Shared GP"), Bonilla et al's [4] batch MTL GP algorithm ("Batch MTGP"), and Ruvolo and Eaton's [23] ELLA with linear base learners. We also considered comparing to the more recent MTL GP method by Rakitsch et al. [21], but their approach cannot handle our problem setting since it requires that each instance be represented in all tasks (with varying labels).

5.1 Data Sets

We used four data sets in our evaluation:

Synthetic Regression Tasks: This set of 100 synthetic tasks was used previously to evaluate lifelong learning [23]. Each task has $n_t = 100$ instances with

$d = 13$ features, with labels that were generated via a linear factorized model on six latent basis vectors with added Gaussian noise.

London Schools: The London schools data set has been used extensively for MTL evaluation. It consists of examination scores from 15,362 students in 139 schools from the Inner London Education Authority. The goal is to predict the exam score for each student, treating each school as a separate task. We employ the same feature encoding as previous studies [14], yielding $d = 27$ features.

Robot Arm Kinematics: We generated this data set by simulating the forward kinematics of synthetic 8-DOF robot arms of various link lengths and configurations. The lengths of each arm link were kept consistent for each task, while the joint angles (all DOFs being revolute joints) varied in each data instance. Each task's goal is to predict the distance of the arm's end-effector from the $\mathbb{R}^3$ point $[0.1, 0.1, 0.1]$. We generated 113 tasks, each with a minimum of 50 joint configurations.

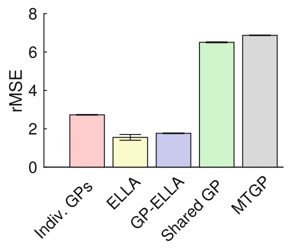
Parkinson's Vocal Tests: This data set consists of vocal signal tests recorded from patients with Parkinson's disease [26]. We split the data set into tasks based on the 42 unique patient IDs and trained on the 16 vocal signal features. The data set had two labels applied to each instance: the linearly interpolated clinician's motor score ("Parkinson's Motor") and the total UPDRS score ("Parkinson's UPDRS"), which we evaluated in separate experiments.

5.2 Methodology

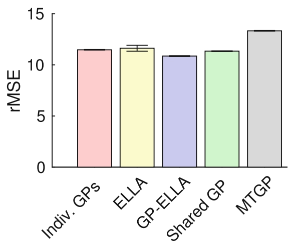
Each data set was evaluated separately. Results were averaged over 10 runs for all data sets. For each trial, the data features and labels were mean-centered, and the data randomly divided into equal-sized training and testing sets. In the parameter verification step, described below, the training set was further divided into a sub-training set and a validation set. The tasks were presented in a random order to the lifelong learners and in batch to the other learners.

The covariance matrix self-noise parameter σ^2 was set to $1.0e{-6}$ for all data sets except London schools, where it was set to 1.0. The variance used in the weight function for comparing $\boldsymbol{\theta}^{(t)}$ values in the prediction step was set to 100 for all data sets. Also, the number of iterations of gradient descent used to learn the covariance function hyperparameters, across all algorithms, was set to 50.

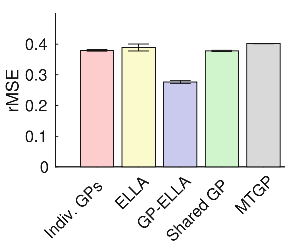
The parameters k, μ, and λ for GP-ELLA and ELLA were tuned on a 50% subset of the training data for the first five tasks, evaluating those parameter values on the validation portion of the training data for those tasks. Then, those tuned parameter values were used for retraining the models for the first five tasks with all available training data, and for learning the remaining tasks. Values of k used in this tuning step were all even numbers from 2 to $2d$, and μ and λ were both set to values in $\{e^{-12}, e^{-8}, e^{-4}, e^{0}\}$. Performance was measured on the held-out test sets after all tasks had been learned.



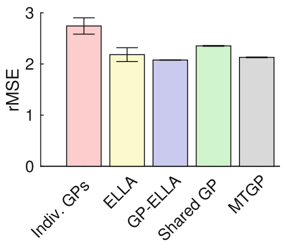
(a) Synthetic



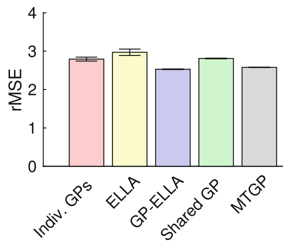
(b) London schools



(c) Robot arm



(d) Parkinson's Motor



(e) Parkinson's UPDRS

Fig. 2. Prediction accuracy reported in root mean squared error (rMSE), where lower scores are better. The whiskers are standard error bars. Our approach achieves consistently superior performance across all data sets, with the sole exception of the synthetic data. On the synthetic data, ELLA is slightly superior since the data was generated using a linear task model.

For Batch MTGP, we use the rank-1 matrix approximation to achieve a training time that is competitive with the other algorithms. However, using this approximation, instead of the more accurate but significantly more computationally intense rank-2 approximation or full-rank matrix (as reported by [4]), caused Batch MTGP to perform worse in some cases than individual GPs.

All experiments were performed on a Mac Pro with dual 6-core Xeon 2.67 GHz processors with 24 threads total; computation for each experiment was limited to four threads. Our algorithm was implemented in MATLAB and makes use of two external packages: MTIMESX, for fast MEX-based computation of the Hessian matrix, and SPAMS, for running LASSO. We also use GPML [22] for GP computations, including minimizing the log-marginal likelihood/fitting the covariance function hyperparameters for a given training data set.

5.3 Results

As shown in Fig. 2 and Table 1, our approach compares favorably to competing methods, including multi-task GP. In terms of prediction accuracy, our method is superior to all competing methods over all data sets, with one exception. The one exception is on the synthetic data that was generated according to a linear task model (the same as used by ELLA), for which ELLA slightly outperforms

Table 1. Computation time in seconds. GP-ELLA shows less than an order of magnitude increase in time over individual GPs, while the Shared GP and MTGP methods show approximately 2–3 orders of magnitude increase in time. Due to its use a linear model, ELLA is the fastest by far, as expected. The standard error of each value is given after the ±.

	Synthetic	London schools	Robot arm	Parkinson's Motor	Parkinson's UPDRS
Indiv. GPs	25.4 ± 0.7	56.9 ± 0.5	10.5 ± 0.2	12.7 ± 0.5	13.1 ± 0.2
ELLA	0.14 ± 0.016	0.22 ± 0.029	0.11 ± 0.002	0.05 ± 0.002	0.05 ± 0.004
GP-ELLA	63.8 ± 1.1	489.8 ± 18.7	38.3 ± 0.2	113.6 ± 4.6	109.4 ± 4.2
Shared GP	$3{,}474.9 \pm 313.7$	$13{,}070.5 \pm 1{,}784.9$	$2{,}023.3 \pm 85.5$	$1{,}972.7 \pm 871.0$	$1{,}150.5 \pm 113.9$
MTGP	$21{,}603.8 \pm 756.5$	$72{,}338.6 \pm 7{,}361.6$	$20{,}124.1 \pm 507.3$	$4{,}449.3 \pm 79.2$	$4{,}538.7 \pm 63.9$

our approach. On all other data sets, GP-ELLA significantly outperforms ELLA, which learns linear models, demonstrating the increased predictive power of non-parametric lifelong learning with GPs.

In terms of computation time, our method is slower than the Individual GPs or ELLA, as expected. Since task-independent GPs are a subroutine of the GP-ELLA algorithm, GP-ELLA will undoubtedly have a higher computation time than the independent GPs, but our results show less than an order of magnitude increase in computation time. Additionally, these results show that GP-ELLA is significantly faster than the competing GP methods, while obtaining lower prediction error. These competing multi-task GP methods are slower than individual GPs by approximately 2–3 orders of magnitude. Please note that the reported computation times do not include parameter tuning for any algorithm, but only the training and evaluation. Our results are consistent across all data sets and clearly demonstrate the benefits of GP-ELLA.

6 Conclusion

Given the recent advances in lifelong learning and batch GP multi-task learning, it is natural to combine the advantages of both paradigms to enable non-parametric lifelong learning. Our algorithm, GP-ELLA, constitutes the first non-parametric lifelong learning method, providing substantially improved predictive power over existing lifelong learning methods that rely on linear parametric models. GP-ELLA also has favorable performance in terms of both prediction accuracy and computation time when compared to multi-task GP methods, with guarantees on convergence in a lifelong learning setting.

Acknowledgements. This research was supported by ONR grant #N00014-11-1-0139, AFRL grant #FA8750-14-1-0069, and AFRL grant #FA8750-16-1-0109. We would like to thank Paul Ruvolo and the anonymous reviewers for their helpful feedback.

Appendix A: Computing the Hessian $\mathbf{H}^{(t)}$

To compute the Hessian $\mathbf{H}^{(t)}$ for the GP loss function, we combine Eqs. 4 and 6. Letting $\mathbf{K}_\sigma = \mathbf{K}_t(\boldsymbol{\theta}^{(t)}) + \sigma_t^2 I$,

$$[\mathbf{H}^{(t)}]_{ab} = \frac{1}{2}\mathbf{y}^{(t)\mathsf{T}}\Bigg(\mathbf{K}_\sigma{}^{-1}\frac{\partial \mathbf{K}_\sigma}{\partial \theta_a}\mathbf{K}_\sigma{}^{-1}\frac{\partial \mathbf{K}_\sigma}{\partial \theta_b}\mathbf{K}_\sigma{}^{-1} - \mathbf{K}_\sigma{}^{-1}\frac{\partial^2 \mathbf{K}_\sigma}{\partial \theta_a \partial \theta_b}\mathbf{K}_\sigma{}^{-1} \tag{12}$$

$$+ \mathbf{K}_\sigma{}^{-1}\frac{\partial \mathbf{K}_\sigma}{\partial \theta_b}\mathbf{K}_\sigma{}^{-1}\frac{\partial \mathbf{K}_\sigma}{\partial \theta_a}\mathbf{K}_\sigma{}^{-1}\Bigg)\mathbf{y}^{(t)} \tag{13}$$

$$+ \frac{1}{2}\mathrm{tr}\Bigg(\mathbf{K}_\sigma{}^{-1}\frac{\partial^2 \mathbf{K}_\sigma}{\partial \theta_a \partial \theta_b} - \mathbf{K}_\sigma{}^{-1}\frac{\partial \mathbf{K}_\sigma}{\partial \theta_b}\mathbf{K}_\sigma{}^{-1}\frac{\partial \mathbf{K}_\sigma}{\partial \theta_a}\Bigg) \ . \tag{14}$$

As an example, consider the squared exponential (SE) covariance kernel function, defined as

$$k_{SE}\Big(\mathbf{x}_i^{(t)}, \mathbf{x}_j^{(t)}\Big) = \sigma_f^2 \exp\bigg(-\frac{1}{2}\Big(\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}\Big)^{\mathsf{T}}\mathbf{M}\Big(\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}\Big)\bigg).$$

If we let $\mathbf{M} = \mathrm{diag}(l_1^{-2}, l_2^{-2}, \ldots, l_d^{-2})$, we now have the ARD variant of k_{SE}. Additionally, if we set $\sigma_f^2 = 1$ and $\boldsymbol{\theta}^{(t)} \triangleq \{\theta_1^{-2} = l_1^{-2}, \theta_2^{-2} = l_2^{-2}, \ldots, \theta_d^{-2} = l_d^{-2}\}$, we have a kernel with $\boldsymbol{\theta}^{(t)} \in \mathbb{R}^d$. Sample values of the first- and second-order derivatives of $\mathbf{K}_\sigma$ are

$$\begin{aligned}
\left[\frac{\partial \mathbf{K}_\sigma}{\partial \theta_a}\right]_{ij} &= \frac{\partial k_{SE}(\mathbf{x}_i^{(t)}, \mathbf{x}_j^{(t)})}{\partial \theta_a} \\
&= k_{SE}(\mathbf{x}_i^{(t)}, \mathbf{x}_j^{(t)})([\mathbf{x}_i^{(t)}]_a - [\mathbf{x}_j^{(t)}]_a)^2\theta_a^{-3} \ , \\
\left[\frac{\partial^2 \mathbf{K}_\sigma}{\partial \theta_a^2}\right]_{ij} &= \frac{\partial^2 k_{SE}(\mathbf{x}_i^{(t)}, \mathbf{x}_j^{(t)})}{\partial \theta_a^2} \\
&= k_{SE}(\mathbf{x}_i^{(t)}, \mathbf{x}_j^{(t)})\bigg(-3([\mathbf{x}_i^{(t)}]_a - [\mathbf{x}_j^{(t)}]_a)^2\theta_a^{-4} \\
&\qquad + ([\mathbf{x}_i^{(t)}]_a - [\mathbf{x}_j^{(t)}]_a)^4\theta_a^{-6}\bigg) \ , \\
\left[\frac{\partial^2 \mathbf{K}_\sigma}{\partial \theta_a \partial \theta_b}\right]_{ij} &= \frac{\partial^2 k_{SE}(\mathbf{x}_i^{(t)}, \mathbf{x}_j^{(t)})}{\partial \theta_a \partial \theta_b} \\
&= k_{SE}(\mathbf{x}_i^{(t)}, \mathbf{x}_j^{(t)})\bigg(([\mathbf{x}_i^{(t)}]_a - [\mathbf{x}_j^{(t)}]_a)^2\theta_a^{-3}\bigg) \\
&\qquad \times \bigg(([\mathbf{x}_i^{(t)}]_b - [\mathbf{x}_j^{(t)}]_b)^2\theta_b^{-3}\bigg) \ .
\end{aligned}$$

References

1. Álvarez, M.A., Lawrence, N.D.: Computationally efficient convolved multiple output Gaussian processes. J. Mach. Learn. Res. **12**, 1459–1500 (2011)
2. Álvarez, M.A., Luengo, D., Titsias, M.K., Lawrence, N.D.: Efficient multioutput Gaussian processes through variational inducing kernels. In: Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 25–32 (2010)
3. Bou Ammar, H., Eaton, E., Ruvolo, P., Taylor, M.E.: Online multi-task learning for policy gradient methods. In: Proceedings of the 31st International Conference on Machine Learning (ICML), June 2014
4. Bonilla, E.V., Chai, K.M., Williams, C.: Multi-task Gaussian process prediction. In: Advances in Neural Information Processing Systems (NIPS), pp. 153–160 (2008)
5. Bou Ammar, H., Eaton, E., Luna, J.M., Ruvolo, P.: Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), July 2015
6. Bou Ammar, H., Tutunov, R., Eaton, E.: Safe policy search for lifelong reinforcement learning with sublinear regret. In: Proceedings of the 32nd International Conference on Machine Learning (ICML), July 2015
7. Chalupka, K., Williams, C.K., Murray, I.: A framework for evaluating approximation methods for Gaussian process regression. J. Mach. Learn. Res. **14**(1), 333–350 (2013)
8. Chen, Z., Liu, B.: Lifelong Machine Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, San Rafael (2016)
9. Deisenroth, M., Ng, J.W.: Distributed Gaussian processes. In: Proceedings of the 32nd International Conference on Machine Learning (ICML), pp. 1481–1490 (2015)
10. Fei, G., Wang, S., Liu, B.: Learning cumulatively to become more knowledgeable. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2016)
11. Flaxman, S., Gelman, A., Neill, D., Smola, A., Vehtari, A., Wilson, A.G.: Fast hierarchical Gaussian processes. Manuscript in preparation (2016). http://sethrf.com/files/fast-hierarchical-GPs.pdf
12. Isele, D., Luna, J.M., Eaton, E., Gabriel, V., Irwin, J., Kallaher, B., Taylor, M.E.: Lifelong learning for disturbance rejection on mobile robots. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2016)
13. Kamar, E., Kapoor, A., Horvitz, E.: Lifelong learning for acquiring the wisdom of the crowd. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI) (2013)
14. Kumar, A., Daumé III, H.: Learning task grouping and overlap in multi-task learning. In: Proceedings of the 29th International Conference on Machine Learning (ICML) (2012)
15. Maurer, A., Pontil, M., Romera-Paredes, B.: Sparse coding for multitask and transfer learning. In: Proceedings of the 30th International Conference on Machine Learning (ICML), pp. 343–351, May 2013
16. Mitchell, T.M., Cohen, W.W., Talukdar, P.P., Betteridge, J., Carlson, A., Gardner, M., Kisiel, B., Krishnamurthy, J., et al.: Never ending learning. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence (2015)

17. Nguyen-Tuong, D., Peters, J.R., Seeger, M.: Local Gaussian process regression for real time online model learning. In: Advances in Neural Information Processing Systems (NIPS), pp. 1193–1200 (2009)
18. Park, S., Choi, S.: Hierarchical Gaussian process regression. In: Proceedings of the 2nd Asian Conference on Machine Learning (ACML), pp. 95–110 (2010)
19. Pentina, A., Urner, R.: Lifelong learning with weighted majority votes. In: Advances in Neural Information Processing Systems (NIPS), pp. 3612–3620 (2016)
20. Quinonero-Candela, J., Rasmussen, C.E.: A unifying view of sparse approximate Gaussian process regression. J. Mach. Learn. Res. **6**, 1939–1959 (2005)
21. Rakitsch, B., Lippert, C., Borgwardt, K., Stegle, O.: It is all in the noise: efficient multi-task Gaussian process inference with structured residuals. In: Advances in Neural Information Processing Systems (NIPS), pp. 1466–1474 (2013)
22. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press, Cambridge (2005)
23. Ruvolo, P., Eaton, E.: ELLA: an efficient lifelong learning algorithm. In: Proceedings of the 30th International Conference on Machine Learning (ICML), June 2013
24. Snelson, E., Ghahramani, Z.: Sparse Gaussian processes using pseudo-inputs. In: Advances in Neural Information Processing Systems (NIPS), pp. 1257–1264 (2005)
25. Thrun, S.: A lifelong learning perspective for mobile robot control. In: Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems (IROS) 'Advanced Robotic Systems and the Real World' (1994)
26. Tsanas, A., Little, M.A., McSharry, P.E., Ramig, L.O.: Accurate telemonitoring of Parkinson's disease progression by noninvasive speech tests. IEEE Trans. Biomed. Eng. **57**(4), 884–893 (2010)
27. Wang, B., Pineau, J.: Generalized dictionary for multitask learning with boosting. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) (2016)
28. Wang, Y., Khardon, R.: Sparse Gaussian processes for multi-task learning. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012. LNCS (LNAI), vol. 7523, pp. 711–727. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33460-3_51
29. Yu, K., Tresp, V., Schwaighofer, A.: Learning Gaussian processes from multiple tasks. In: Proceedings of the 22nd International Conference on Machine Learning (ICML), pp. 1012–1019 (2005)