Online Knowledge Level Tracking with Data-Driven Student Models and Collaborative Filtering

Antoine Cully and Yiannis Demiris, Senior Member, IEEE

Abstract—Intelligent Tutoring Systems are promising tools for delivering optimal and personalised learning experiences to students. A key component for their personalisation is the student model, which infers the knowledge level of the students to balance the difficulty of the exercises. While important advances have been achieved, several challenges remain. In particular, the models should be able to track in real-time the evolution of the students’ knowledge levels. These evolutions are likely to follow different profiles for each student, while measuring the exact knowledge level remains difficult given the limited and noisy information provided by the interactions. This paper introduces a novel model that addresses these challenges with three contributions: 1) the model relies on Gaussian Processes to track online the evolution of the student’s knowledge level over time, 2) it uses collaborative filtering to rapidly provide long-term predictions by leveraging the information from previous users, and 3) it automatically generates abstract representations of knowledge components via automatic relevance determination of covariance matrices. The model is evaluated on three datasets, including real users. The results demonstrate that the model converges to accurate predictions in average 4 times faster than the compared methods.

Index Terms—Student Model, Knowledge Level Estimation, Machine Learning, Gaussian Processes.

1 INTRODUCTION

Intelligent Tutoring Systems (ITS) have the potential to deliver tremendous benefits to education by providing each student with an optimal and personalised learning experience. ITS have received a significant amount of attention for at least four decades [1], [2], [3]. Current ITS have proven to be nearly as effective as human tutors [4] and are already in use in several schools across the world [5]. As examples, the ASSISTment system was used by 600 students for one year as part of their maths classes [6], while the Cognitive Tutor mathematics courses [7] are in use on a weekly basis by 600,000 students across 2600 schools each year, having demonstrated more efficient learning when compared to traditional courses [8]. In addition to these direct pedagogical effects, ITS are also used to help children with asthma or diabetes to learn more about their disease leading to therapeutic benefits as well [9].

In order to use and generate personalised content for each student, ITS are in general composed by several modules [7] that each plays a specific role: 1) a task selection module, 2) a cognitive model, 3) a hint factory, and 4) a student model. The role of the task selection module is to select the next task or question that will be presented to the student according to an estimate of his/her current knowledge level; subsequently the response (or action) of the student is evaluated by the cognitive model, which hypothesises the reasoning performed by the student to produce the given answer. When necessary, this estimated reasoning is then used to generate hints that are personalised according to the cognitive process assumed to be followed by the student [10]. Finally, the answer of the student is used to update the student model that tracks the current knowledge level of the student on different knowledge components. The knowledge level of a student is typically defined as the probability of them correctly answering a question [7], [11]. Each question belongs to one of the knowledge components that the student attempts to master. The different knowledge components of a domain are usually manually defined by the teachers. Automatic methods can also be used to extract them from textbooks [12] or from online student discussions [13].

However, an ITS does not necessarily contain all these...
modules. For instance, the task selection module can be designed to select directly the next tasks without the need of a specific student model. This approach has been used in [14], where the task selection module selects the next activity by maximising the learning progress of the student, using a multi-armed bandit algorithm.

Similarly, the cognitive model and the hint factory are often directly defined according to the application domain, and are frequently difficult to transfer to other educational domains. For instance, the vast majority of deployed ITS are applied to maths or physics problems [15], [16]. These domains are particularly interesting for cognitive models and hint factories because the link between the answer given by the student and the reasoning he/she employed can be estimated with high certainty under specific conditions. For example, it is relatively simple to detect if, in a mathematical operation (e.g., an addition or a multiplication), the student forgets to take into account a carried out number. Such relationships are more difficult to define in other application domains.

In the case of diabetes related questions, which are considered in the applications of this work, a link between wrong carbohydrate counting and a potential reasoning error is more difficult to establish. For this reason, several ITS do not rely on a cognitive model or a hint factory and are thus only composed of the task selection module and the student model [17]. Most of the time in this context, the objective of the systems is to select a question that the student model predicts to be sufficiently challenging for the student but also not above his/her capabilities. This objective is directly linked to the Zone of Proximal Development which is known to favour learning [16], [18]. Other selection approaches rely on the relationships between the knowledge components to select questions that the students has to master before considering more advanced questions [19].

Several studies have demonstrated that using personalised student models provides critical benefits for the students, compared to the use of a generic student model fitted on the whole population of students [20]. However, designing a personalised student model able to rapidly adapt to the specificities of a student and to track his/her knowledge level on several knowledge components remains a challenge. The difficulty relies in the fact that each answer of the student contains very limited amount of information about the knowledge of the student on a specific knowledge component, as it only informs the system that the student managed to correctly/incorrectly answer the question. In order to capture the level of knowledge of the student, the system needs to ask several questions. Moreover, when we need to predict the knowledge level on several knowledge components, the amount of interactions required to make an accurate prediction becomes too large to expect a rapid adaptation and personalisation of the system. Moreover, the personalisation of the model is often produced as a post analysis and is rarely done in an online fashion [7]. The reason for this is that personalising the model requires a significant amount of data, which is only available after extended periods of interaction with the system.

In this paper, we present a novel student model called GPCF for Gaussian Process with Collaborative Filtering, which combines three contributions (see Fig. 1): 1) the student model can be trained online and can track the changes of the student’s knowledge level over time, 2) it leverages the information provided by previous users of the system to rapidly provide accurate predictions even when few data is available (a problem often named bootstrapping, or cold-start problem [21]), and 3) it automatically generates an abstract (or latent) representation of the knowledge components according to their mutual influence. In particular, this captures how making progress on one knowledge component can lead to progresses on other components. In practice, these contributions allow GPCF to provide real time personalised predictions even when limited data is available and progressively refine them as more data is collected. For instance, this is crucial when designing new courses for a small cohort of students. In this kind of situations, GPCF is more accurate than methods from the state-of-the-art, including deep neural networks [22], and converges to accurate predictions in average 4 times faster than the compared methods.

The overall motivation behind this model is to allow ITS to be effective even when few data is available; examples being a new class or a new syllabus. Most of current ITS systems use datasets of several hundred students interacting with the systems several thousands of times. This makes applying these models on a domain where limited data is available difficult. In this paper, we show how GPCF can be used without prior data or with small datasets of only a handful of prior students. To achieve this, GPCF relies on its own experience, similarly as a human tutor: initially, when little data has been recorded about a new student, it is safer to assume that he/she is behaving like one of the previous students, then when more data has been collected, the tutor can progressively take into account the particular needs of the new student.

2 RELATED WORK

2.1 Student models

A large number of different approaches to create student models have been introduced during the last few decades. A thorough review is provided in [3], where the authors categorise the different models depending on the approach/techniques employed and the user characteristics selected for model creation (e.g., machine learning algorithms, stereotypes, overlays). In this paper, we focus on the machine learning based approaches, as they are the most relevant to the proposed student model. Three types of student models are mainly used in the literature and are detailed below.

Bayesian Knowledge Tracing

The first one is the Bayesian Knowledge Tracing (BKT) approach [11] that represents the student as a two-state Hidden Markov Model (HMM). It encodes the probability of a student responding correctly to a question (or task), conditioned on whether or not he/she has mastered the knowledge component. This model also takes into account the probability of a student failing to answer the question because of “slip” (inattentive errors) or that he answers correctly thanks to a lucky guess. These probability values are defined by the parameters of the model, which need to
be fitted to the collected data. Initially, the parameters of this model have been fitted to large groups of students in order to build generic student models for all the students [11]. Several approaches have been used to fit the parameters of the model, including curve fitting algorithms [11], Expectation Maximisation algorithm [23], and grid search [24]. Following a similar approach, the authors of [25] use a Partially Observable Markov Decision Processes to model the mental state of students and generate a policy that is capable of identifying their knowledge gaps.

However, several studies later revealed that fitting the model to each individual student provides better personalisation of the system, and thus, a better overall learning experience [20], [26]. In particular, these works demonstrated that the parameters found vary significantly from one student to another, which advocates the need for personalised student models. The model introduced in this paper is designed to provide such personalised predictions.

Several studies also demonstrated that the BKT model is difficult to fit (and thus to personalise) as it suffers from an issue called the “identifiability” problem that may make the parameters of the model implausible [27]. For instance, the probability of a successful guess can be larger than the probability of answering correctly questions related to a mastered knowledge component [27]. This problem is amplified when the training datasets are small, which makes personalization of the model more difficult. In order to cope with these issues, several improvements of the model have been proposed. These approaches use for instance, Dirichlet priors to bias the model search [27], [28], or constrained grid search to generate plausible configurations [29].

Performance Factor Analysis

The second well known approach for designing student models is the Performance Factor Analysis (PFA) [30], [31]. This model is based on a single parameter logistic function that corresponds to an elaboration on the Item Response Theory [32]. This model predicts the probability of the student correctly answering a question related to a knowledge component, according to the weighted sum of prior successes and failures on questions from the same knowledge component, coupled with a general difficulty bias. Each knowledge component is linked to a different PFA, with independent parameters, which are fitted to each knowledge component.

Deep neural networks

More recently, a new family of knowledge tracing algorithms using (deep) recurrent neural networks have been proposed [22], [33]. In particular, the Deep Knowledge Tracing (DKT, [22]) algorithm traces the knowledge of students using a Long Short Term Memory (LSTM) recurrent neural network. After each question, the model predicts the probability of successfully answering the next exercise. This algorithm has also been extended with a memory network to improve its prediction accuracy [33] and has been extensively compared to existing algorithms in the literature [34]. The experimental results demonstrate that DKT consistently leads to state-of-the-art results across most datasets [34]. Several recent variants have also been proposed in order to embed additional features about the students in DKT. For instance, prerequisite relations between the knowledge components [35] or manually defined features [36], can be integrated in DKT to improve its performance. This model can also be used to automatically create groups of students according to their abilities [37].

Other types of student model

In addition to these three predominant models, other approaches have been introduced. For instance, the authors of [38] suggest a generic process, which takes as input several sources of information about the student and the context in order to predict whether the next answer of the student will be correct and how much time is required for their response. The suggested sources of information are: 1) the student’s level of proficiency, 2) the difficulty of the knowledge component, 3) the difficulty of the question, and 4) the number of hints that the student has seen or general information about the context. While this approach can be used with a large variety of learning mechanisms (e.g., Bayesian classifiers, decision trees, and linear regression [38]), it requires a significant amount of information about the context and the problem, which are not systematically available, like the student’s level of proficiency.

Another approach to model the proficiency of a student on a specific knowledge component is to use a rating system. For instance, the Glicko rating system has recently been used to assess the level of a student, but also the level of each question or task of the knowledge component [39]. The Glicko rating is commonly used to rank players in chess, Go, and many other domains, and captures the probability of a player to win against another player according to the difference of rating. After each confrontation, the rating of each player is adjusted (the winner’s rating increases while the loser’s decreases). By rating both the students and the questions, it is therefore possible to estimate the probability of the student to answer correctly to a question according to their respective ratings. For instance, Schadenberg et. al. use the Glicko rating system to select the next questions to be presented to a student so that the student’s probability of success remains around 70% [17]. This selection strategy is designed to allow the student to stay in his/her zone of proximal development [18]. In the experimental evaluation, we will compare the proposed model with Glicko and DKT, as they are one of the few models in the literature that track the knowledge level of the users in an online fashion over time.

In the last few years, new models have been proposed to improve their predictions based on new aspects of the users [40]. For instance, the affective states of the students can be taken into account in order to improve the overall interaction with the ITS [41]. Another very active domain of research is to design ITS and student models for persons with cognitive impairments [42]. For instance, ITS and student models can be used to assist students with autism in their first inclusion during mainstream classrooms [43].

Combinations of multiple models

While there is a large variety of student models and training procedures, it is particularly difficult to tell which one offers the best predictions. Several contradicting studies have claimed that approaches, such as BKT or PFA, perform
better than their counterparts [29], [30], [44], [45], [46]. In order to address these contradicting works, Baker et al. [44] combine several student models (like BKT and FPA with a few others) in an ensemble learning algorithm [47] that merges the different predictions. Unfortunately, the authors report that they found no evidence that ensemble models perform significantly better than the best individual models. Conversely, a recent study proposed to combine the BKT and FPA models into a single hierarchical Bayesian model, which outperforms the two underlying models when considered separately [48].

**Current limitations**

All of the approaches (except DKT and Glicko) presented so far are “model-based” methods, which means that the student model is based on a predefined function or model. For instance, the BKT is based on a two-state HMM, while the FPA is based on a logistic function. This is a strong assumption that has rarely been addressed in the relevant literature. For instance, using a logistic function implies that the students will not experience intermediate “learning plateaus” before reaching their maximal performance; however, we can expect that each student will likely experience a different learning curve, potentially containing several learning plateaus.

Moreover, the personalisation of the models is mostly done as a retrospective analysis [7], [20], and being able to personalise the student model in an online fashion remains an open challenge [7] (Glicko and DKT being exceptions [39]). Furthermore, the model needs to take into account the progress of the student when the ITS is used for an extended period of time. For this reason, the model needs to be updated and fitted again periodically in order to track changes in the student’s proficiency.

In order to tackle these two fundamental challenges, we present in this paper GPCF, a student model which does not constrain the evolution of the knowledge level (this type of approach is often called data-driven), and which is able to track its evolution on different knowledge components over time in an online fashion. The main difficulty with tracking the knowledge level in real time is that it requires a sufficient amount of data to make accurate predictions. To address this issue, we combine our model with a collaborative filtering approach that allows the proposed model to leverage the information collected about previous users of the system to make accurate predictions after only a small number of iterations with the student.

### 2.2 The Cold-start problem

Providing accurate estimates of the level of proficiency of a new student given a limited amount of data is similar to the problem of providing accurate estimates of preferences for new users in recommendation systems, like on Amazon or Netflix. These systems suggest items or movies to new users, even if they initially have little information regarding the users’ interests. In this domain, the challenge of estimating user preferences without a priori user information is particularly well documented as the “cold start” problem, and two main approaches have emerged: Content-based recommendations and Collaborative recommendations [21].

**Content-based recommendations**

In the first case, the main concept consists of finding items that are similar to those that the user appreciated in the past. This is achieved by using similarity measures on the attributes of the items [49]. For instance, if items A and B are similar, then it is likely that users who liked item A will like item B as well. The concept can be also centred around users: if two users are similar (e.g., according to their ages, origins, backgrounds), then they are likely to share the same preferences. Unfortunately, this approach requires access to a certain amount of prior information about the user or item in order to be able to compute similarity measures, which is often not the case in ITS scenarios. Typically, in the scenarios considered in this paper, only the name of the different knowledge components is known in advance. Moreover, while demographic information of the students contains a significant amount of insight for improving recommendations [3], [50], the link between these attributes and the knowledge level is not consistent. For instance, the age of the students does not always reflect their level of knowledge, as they might be at different stages of development. This is most likely to happen in domains of knowledge that are not taught at school. For instance, the knowledge level related to diabetes can be uncorrelated with the age of children, as the diagnosis of the disease may happen at different periods and that the social environments may be different.

**Collaborative recommendations**

The second approach, collaborative recommendation (also named collaborative filtering), uses information collected from previous users (for example, how much they rated a particular item) to base its predictions for a user who has never encountered this item, or for a new item never presented to the community [51]. For instance, if a group of users like both items A and B, it is likely that a new user who likes item A will also like item B. This approach can be used in the context of student modelling, as we can hypothesise that if several students have similar levels of proficiency across several knowledge components, then it is likely that their knowledge level is similar for other components. To the best of our knowledge, there is only one application of collaborative filtering in the domain of student modelling. This approach consists of automatically defining the parameters of a logistic function (Item Response Theory) to fit the model of the student responses over different items [52].

The GPCF model introduced in this paper is based on this idea of collaborative filtering. Each student is modelled with a temporal, data-driven predictor, a Gaussian Process (GP), which can combine predictions from other student models to make accurate predictions, even with a limited amount of data. The next section presents the technical details of this model.

### 3 Methods

The objective of GPCF is to predict the knowledge level of the student simultaneously across several knowledge components. This objective presents three challenges: 1) the knowledge level is likely to change over time, as the student
is progressing; 2) these changes are unlikely to follow the same evolution for every student; 3) the actual knowledge level cannot be directly measured, as the interactions with the system only capture whether the student answered successfully a question (binary information). This last challenge requires averaging the result of several questions to obtain an accurate estimate of the student’s level of proficiency in one knowledge component. The problem is further complicated by an increase in the number of components. GPCF aims to tackle these three challenges by using a data-driven regression algorithm that tracks changes over time, combined with a collaborative filtering approach that allows the model to make accurate predictions after a limited amount of interactions with the system.

3.1 A temporal data-driven student model

The primary goal of ITS is to enable the student to increase his/her knowledge while interacting with the system. In order to adapt the difficulty of the questions to the student, it is necessary to adjust the predictions of the student model over time to reflect variations in the student’s level of proficiency. Moreover, as explained above, the evolution of this proficiency level is likely to follow a different path for each student. For instance, in the case of diabetes-related questions (which are considered in our experiments), each child has his/her own prior knowledge or interest in learning more about his/her condition. For this reason, the student model introduced in this paper is based on a data-driven approach, which means that it does not rely on a pre-defined model or function to explain the progress of the child, but rather allows arbitrary shape for the evolution of the knowledge level.

3.1.1 Gaussian Processes

GPCF is based on GPs, which is a statistical data-driven approach that can be used as a non-linear regression algorithm [53]. It uses data collected in a continuous space (which could be a spatial and/or temporal domain) and predicts the value of the underlying unknown function at locations where no data has been collected yet. However, instead of predicting a single value (similarly to most regression algorithms), GPs predict the probability distribution of the possible values at each point of the domain. This statistical representation provides information about the confidence of the predictions. For instance, the probability distribution is narrow when the uncertainty about the prediction is low, and wider when the uncertainty is high (see Fig. 2). In this paper, we use the regression abilities of the GP to adjust the prediction over time and its statistical nature to automatically extract the knowledge level (success rate) of the user.

**Notations:** The correct or wrong answers of the student are collected into a set of (binary or continuous) observations \( P_{1:t} \) and are used to compute the GP as follows:

\[
P(f(x)|P_{1:t}, x) = \mathcal{N}(\mu(x), \sigma^2(x))
\]

where:

\[
\mu(x) = \mu_0 + k^T(K + \sigma^2_{\text{noise}}I)^{-1}(P_{1:t} - \mu_0)
\]

\[
\sigma^2(x) = k(x, x) - k^T(K + \sigma^2_{\text{noise}}I)^{-1}k
\]

where \( \mathcal{N} \) denotes the standard normal distribution. \( \mu \) is the mean function of the GP, \( \sigma \) is the corresponding standard deviation, \( x \) refers to a point in the input space, which denotes in this paper a one-hot encoding of the considered knowledge component and the relative time when the interactions with the system occur (more details are provided in the next section).

GPCF aims to model the propensity of the student responding correctly to questions on each of the knowledge components of an ITS (e.g., being able to perform additions, or multiplications), and to track the variations of these success rates over time. For this purpose, we use the regression abilities of GPs to take into account the moment at which each data point has been recorded in order to adapt the predictions over time.

In this paper, we propose to consider all the knowledge components in the same GP by using multi-dimensional inputs (one dimension per knowledge component [53], [55]). The advantage of this approach is that it can exploit potential correlations between the different knowledge components. For example, if two knowledge components are correlated, then when a student makes progress in one of them, the model can predict that he is also very likely to improve in the other component. A global GP can capture this correlation and can predict simultaneous variations in the knowledge level. More details are provided in section 3.3.2. The drawback of this approach is that it makes training the GP computationally more expensive. In our experiments we use several dozens of knowledge components and students interacting with the system over an extended period of time, yet the computational cost of this approach has never been a limiting factor.

The temporal aspect and the different knowledge components are taken into account by defining the input space of the GP (which defines \( x \)) with a dimensionality equal to the number of knowledge components plus one dimension for the time. For instance, if the ITS contains two knowledge components, then the input space has three dimensions. The first dimension of \( x \) defines the time and consists of the correct or wrong answers of the student.

![Fig. 2](image.png)
of a counter that is incremented after each interaction or question. The other dimensions of \( x \) are used to define the considered knowledge component in a binary fashion (one-hot encoding): the dimension associated with this knowledge component is set to 1, while the other dimensions are set to 0. Using different dimensions for each knowledge component enables the predictions to be initially independent, as the distance in the input space between points associated with different knowledge components is large enough so that the GP considers no correlations between these inputs. After recording enough data points, an automatic relevance determination procedure (detailed in section 3.3.2) enables GPCF to adjust the scale factor between different dimensions in order to leverage the correlation between the knowledge components.

An alternative approach could have been to consider a different GP for each of the knowledge components, each of them being used to predict the success rate of the student in a specific knowledge component. This allows the training of the models to be quicker, as there is less data to process per GP and the training process can be performed in parallel. However, it prevents the model from taking advantage of potential correlations between the different knowledge components.

Fig. 2 shows how considering the time of acquisition as the input space allows the model to capture changes in the user’s knowledge level over time. For this illustration, we used an arbitrary function containing several learning plateaus in order to highlight the ability of the GP to adapt to the personal development of the students.

3.1.3 Prediction of the success rate given binary observations

The second difficulty in ITS is that the collected data does not represent the actual knowledge level or success rate of the user. The information provided by the system greter each quiz question defines only whether the student managed to respond correctly to the question or not. For this reason, the student model needs to be able to deal with binary information in order to continuously predict the success rate. To achieve this, we used the statistical nature of the GP to combine recent data points by considering each of them as really uncertain (i.e., with a large acquisition noise). This is achieved by using a particularly large value for the acquisition noise (\( \sigma_{\text{noise}} \)). In the experiment presented in this paper, this parameter is set to 1, while common values are situated around \( 10^{-3} \) (see section 4.1 for the hyper-parameters values). This property, which is rarely used in the literature, allows our student model to both change its predictions over time and to aggregate binary data into a success rate. Fig. 3 illustrates these properties.

Another important property of GPCF is its ability to deal with periods of time where no data is recorded. This is likely to happen when the user does not interact with the ITS during extended periods of time. After such periods of inactivity, it is very likely that the knowledge level of the student has changed. The proposed student model automatically takes this into account by increasing the uncertainty of its predictions, while making the predictions slowly closer to the mean term (\( \mu_0 \)). This is an inherent property of GP when data are sparse in certain regions of the input space. This property is illustrated on Fig. 4.

3.2 Data-efficient collaborative filtering for knowledge level estimation

Another challenge in the context of ITS is the amount of data required to train the model in an online fashion in order to
To achieve this objective, we combined collaborative filtering with a gradient-based optimiser (Rprop) to provide to a robot with prior knowledge about the different behaviours it should expect from a new user. Similarly to the approach in [56], we use a “behavioural map” to provide a prior distribution of the model. For instance, in Fig. 3, several data points are required to update the predictions. In practice, this means that the student has to answer dozens of questions before a traditional algorithm can collect enough data to make accurate predictions. This also means that during this period, the student could not benefit from personalised predictions. Therefore, the second objective of GPCF is to minimise the amount of data required to make accurate predictions. To achieve this objective, we combined collaborative filtering techniques with the student model described above. The main idea of our approach is to use the data collected from the previous users of the system in order to make more accurate predictions after a small number of interactions. This is achieved by using the data collected with the current student to determine which of the previous users has a model that explains most of the observed data (also named the likelihood function). With this approach, we do not need to compare attributes of the students because it only relies on the predicted level of proficiency.

We integrated this approach in GPCF by substituting the mean term of the GP ($\mu_0$) with a fusion operator, which combines predictions from the previous users. The mean term is used to define the output of the model when there is no data. For example, in Fig. 2, the mean term is defined as a constant value. However, arbitrary functions can be used as a mean term in a GP [53]. For instance, the mean term has been replaced by a “behavioural map” in [56] to provide to a robot with prior knowledge about the different behaviours it could employ to face unanticipated situations.

In this paper, the fusion operator is a linear combination of the predictions coming from the previous students. The advantage of using a fusion operator in place of the mean term is that the mean term provides the global shape of the prediction while the rest of the GP encodes the local specificities of the new student. For instance, if the new student is very similar to a previous user across most of the knowledge components, but very different in one of them, the fusion operator will suggest to use predictions from the previous user as a first guess, while the GP will progressively refine the predictions by taking into account the student’s differences. Fig. 5 illustrates the general structure of the student model used.

From a mathematical point of view, GPCF with the linear combination is obtained by substituting the term $\mu_0$ from equation 1 with the linear combination $O(x)^T W$:

$$P(f(x)|P_{1:t}, x) = N(\mu(x), \sigma^2(x))$$

with:

$$\mu(x) = O(x)^T W + k^T(K + \sigma^2_{\text{noise}}I)^{-1}(P_{1:t} - O(\chi_{1:t})^T W)$$

$$\sigma^2(x) = k(x, x) - k^T(K + \sigma^2_{\text{noise}}I)^{-1}k$$

$$O(x) = \begin{bmatrix} \mu_{c_1}(x) \\ \mu_{c_2}(x) \\ \vdots \\ \mu_{c_N}(x) \end{bmatrix} ; \ W = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_N \end{bmatrix}$$

$$O(\chi_{1:t}) = [O(\chi_1) \ O(\chi_2) \ \ldots \ \ O(\chi_t)] ;$$

(2)

Where the vector $O$ gathers the predictions from the student models of the $N$ other users and where the vector $W$ contains the weights used for the fusion of the different predictions. The vector $\chi_{1:t}$ contains the input values ($x$) corresponding to each recorded data point ($P_{1:t}$). In this paper, we only consider a simple fusion operator (i.e., a linear combination). However, more complex fusion operators, such as neural networks, can also be considered.

3.3 Automatic relevance determination

3.3.1 Fusion operator

The parameters of the fusion operator are automatically determined by optimising the “marginal likelihood” of the model given the data [53]. In our case, the weights $W$ of the linear combination are determined with this procedure at the end of each quiz session (more detail in the experiment section). The algorithm automatically determines the combination of predictions that best matches the observed data. The combination of the fusion operator and the likelihood optimisation of its parameters provided by the GP leads to an adaptive collaborative filtering method. Similarly to the collaborative filters in recommendation systems described in Section 2, GPCF automatically finds the previous user (or the combination of previous users) that explain the most the data collected from a new user.

The marginal log likelihood of the model is defined as:

$$\log P(P_{1:t} | \chi_{1:t}, W) = -\frac{1}{2} A^T(K + \sigma^2_{\text{noise}}I)^{-1} A - \frac{1}{2} \log |K + \sigma^2_{\text{noise}}I| - \frac{t}{2} \log 2\pi$$

where $A = P_{1:t} - O(\chi_{1:t})^T W$

(3)

The optimisation of the marginal log likelihood is done with a gradient-based optimiser (Rprop [57]). We can observe that the inverse of $(K + \sigma^2_{\text{noise}}I)$ only needs to be computed once as it does not depend on $W$. Thanks to that the evaluation of the marginal log likelihood is not computationally expensive and can be performed in real-time.

3.3.2 Kernel function

In addition to the parameters of the fusion operator, the automatic relevance determination can also be used to define
those of the kernel function \( k(\ldots) \). This process enables the student model to autonomously find correlation between the different knowledge components. For example, if making progress in one of the components also leads to progress in a second component, the model is able to find and leverage these correlations.

The kernel function used in our experiments follows the form:

\[
k(x_1, x_2) = \sigma^2 \exp\left(-\frac{1}{2}(x_1 - x_2)^T \Delta x (x_1 - x_2)\right)
\]

where \( \Delta \) represents a triangular matrix (with positive elements on the diagonal), and is used to form a covariance matrix via the \( \Delta^T \Delta \) product. The coefficients of the first column of \( \Delta \) are fixed to zero, except for the first value of the column which is set to \( \sigma_{\text{time}} \), an arbitrary user-defined defined value that controls the variation speed of the knowledge level captured by the model (see appendix 4.1). The other coefficients of \( \Delta \) can be tuned to maximise the marginal log likelihood defined in the previous section. The optimisation of this function allows the model to determine if certain correlations between the different knowledge components may explain the observed data. For more detail, we refer interested reader to [58].

The generated covariance matrix can be used to analyse the knowledge component correlations found by GPCF. In particular, an Eigen-Value decomposition of the matrix provides information about the main combinations of components (similar to a principal component analysis). With this analysis, it is possible to define abstract knowledge components, which regroup several components according to their correlations. This ability is investigated in the next section.

4 EXPERIMENTAL VALIDATION

We evaluate the ability of GPCF to predict the knowledge level of the students on three datasets and we compare its performance against two approaches from the state-of-the-art. The accuracy of the models are assessed based on the area under the curve (AUC) metric as it is customary in the literature [22, 33, 34] or, when available, on the ground truth knowledge level. The convergence speed (i.e., the number of data points required to make accurate predictions) is evaluated under two conditions: 1) stationary, and 2) non-stationary knowledge level. This evaluates the ability of the models to track changes over time. Finally, the ability of GPCF to extract an abstract representation of the knowledge components is demonstrated using data from real users. Details about the compared approaches and the datasets are given in the following sections.

4.1 Compared approaches

We consider two algorithms from the state-of-the-art as a comparison against GPCF:

Deep Knowledge Tracing (DKT): This approach uses a Long-short term memory (LSTM) deep neural network, to predict the knowledge level of the students [22]. Based on the source code provided by the authors of DKT, we re-implemented the DKT architecture in TensorFlow and replicated the reported performance on the same dataset, as indicated in Table 1.

Glicko: This approach originates from [17], which uses the Glicko rating system [59] to assess the level of the user compared to the difficulty of the topics. The hyper-parameter values are initialised according to the specifications of [17] (see appendix 4.1 for the hyper-parameter values). The difference in Glicko rank is used to define a probability of success (i.e., responding to the question correctly).

For further analysis, we also consider three variants of the GPCF model to evaluate the contribution of each of its components.

Data-only: This approach uses the data collected from a new user to compute the average success rate for each topic. For the topics in which no data has been collected, the predicted success rate is 50%, otherwise the success rate is the number of correct answers divided by the number of questions asked. An averaging window using the last 20 data-points for each knowledge component is also considered in the second experiment.

CF-only: This approach is "Collaborative Filtering-only". It searches for the previous user with the nearest success rate compared to the predictions made by the "Data-only" method, and uses the success rate of this student as a prediction of the success rate for the new user.

NOCF: This approach does not use the collaborative filter of the GPCF model and only uses the predictions made by the GP. This variant corresponds to the case of using the proposed model without prior users (i.e., with no training dataset).

Hyper-parameter values

The hyper-parameters for GPCF and for the Glicko model are detailed in the following table. GPCF uses the same hyper-parameter values for all the experiments except for the KDD dataset (in which it uses 100 for \( l_{\text{time}} \)) because of the automatic generation of abstract concepts. Note: \( l_{\text{topic}} \) is only used to initialize the diagonal of \( \Delta \) before the automatic relevance determination (see section 3.3.2). The values for the Glicko Model come from [17].

<table>
<thead>
<tr>
<th>Proposed Model</th>
<th>Glicko</th>
</tr>
</thead>
<tbody>
<tr>
<td>( l_{\text{time}} )</td>
<td>( \sigma^2_{\text{noise}} )</td>
</tr>
<tr>
<td>50 (100)</td>
<td>1</td>
</tr>
</tbody>
</table>

4.2 Datasets

We use three different datasets:

DKT dataset: We use the same dataset as in the original DKT paper [22]². This dataset is composed of 4000 virtual students (2000 for training, and 2000 for testing) who answered 50 exercises drawn from five concepts. The exercise order is the same for all the students. Each student has an unknown knowledge state for each concept, which increases over time in order to simulate his progress. Like in the original paper, the experiment is replicated 20 times with different randomly generated data by using the different datasets provided by the authors.

Because our model assumes that several exercises for the same knowledge concept should happen in order to estimate

1. dataset available at this address: https://github.com/chrispiech/DeepKnowledgeTracing
the student’s knowledge level, we changed the encoding of
the dataset to be a one-hot encoding of the concept (and not
of the exercise index as originally defined). We did not notice
any performance drop from DKT caused by this change
(AUC of 0.75 with the original encoding, versus 0.76 with
the concept-based encoding).

In order to evaluate the influence of the number of prior
users (i.e., the size of the training set), we evaluated the
models on different variants of this dataset containing 2000,
100, 20, and 10 students (we keep the N first students of
the original training set). When using these datasets, GPCF
updates the parameters of its fusion operator every five
questions.

**PAL dataset**: The second dataset uses data collected
from 27 children who interacted with a quiz game developed
for the European Project PAL (Personal Assistant for an
healthy Lifestyle). The quiz game contains 36 knowledge
components (e.g., knowledge about nutrition, glucose, in-
sulin injection) and was designed to allow users to improve
their knowledge about diabetes and the corresponding
treatments. The data was collected daily from the children
over a period of 2 months. Among the participating children,
we have selected the 27 ones who answered more than 10
questions per knowledge component on average. With the
collected data, we computed the success rate of each child
on the different knowledge components of the quiz game.

In order to simulate children who interact with the system
while knowing the “ground truth” of their knowledge level,
we use the success rates of the real children to generate
virtual children (as suggested by [60]). These virtual children
respond to the quiz questions with the same (constant)
success rate as the real children. For the evaluation, one
of the 27 virtual children is used as the “new student” and
is considered as unknown, while the 26 others represent
the previous students that have interacted with the system
during 2000 time steps (corresponding to 2000 questions).
When using this dataset, GPCF updates the parameters of its
fusion operator every five questions.

In order to study the ability of the models to track changes
in the knowledge level that happen over time, we created a
“dynamic” variant of this dataset in which we simulate
the evolution of the knowledge level by using a logistic
function with randomly generated parameters for each of
the knowledge components (see Fig. 7). The parameters
are defined so that the evolution converges towards the
success rates recorded with the real children, while the initial
performance of the virtual child is defined as a random
fraction of the final success rate (between 10% and 75%),
the mid-point of the progression is set between 50 and 75
interactions, and the steepness of the curve between 1.5
and 2.5. For this experiment, the parameters of the fusion
operator of GPCF are updated every 25 interactions, while
the total duration of the experiment has been multiplied
by five. Details about the datasets are given in Table 1.

The update the GPCF fusion parameters is done every 25
questions.

The main difficulties associated with this dataset are
that (1) data recorded a long time ago are not relevant to
predict current knowledge levels, and (2) the success rate
may change rapidly making the tracking particularly difficult,
as the amount of relevant data becomes relatively small. It

is also important to note that the random generation of the
temporal evolution makes it less likely to have correlations
between the virtual users.

The experiments are replicated 27 times in order to gather
statistics about the quality of the predictions. Each of the replications uses a different virtual child and the 26 remaining
ones as the previous students. This “leave one out” approach
to replicate the experiment is often used as a cross validation
method. While these experiments could have been conducted
with fully randomly generated success rates for the virtual
children, the use of data from real children demonstrates that
the collaborative filtering approach is able to rapidly extract
meaningful relationships between the previous students
without artificially adding correlations to their data. In this
dataset, like in the previous one, the automatic relevance
determination of the kernel hyper-parameters of GPCF is
disabled, as correlations between the knowledge components
are artificially added when generating this synthetic
dataset. For instance, if two components have the same
success rate then this dataset will consider them as correlated
for no reason, which might bias the results. However, the
automatic determination of the kernel hyper-parameters is
activated in the next dataset, as it exclusively uses data from
real users.

**KDD dataset**: The last dataset used in this paper
is the “Algebra 2005-2006” dataset from [61]. It contains
813,661 interactions from 575 students who used an ITS for
mathematics lessons. We filtered the dataset to keep only the
students who answered more than 2000 questions in total on
the 10 most frequent knowledge components (51 students in
total). After filtering, the dataset contains for each student a
list of their responses to the maths questions, which states
whether the student responded correctly (in a first attempt)
to the question, and to which knowledge component this
question is associated. 20 users were used as prior users (i.e.
the training set), while the 31 others are used for testing.

With this dataset, the update the GPCF fusion parameters is
done every 25 questions.

### 4.3 AUC Results

Table 1 shows the AUC values for all the compared datasets
and approaches. The table also reports the values for the
Bayesian Knowledge Tracer (BKT, [11]) extracted from [22].
The results on the DKT dataset show that most of the
compared approaches perform well when a large training
dataset is available. For instance, when 2000 students with
50 interactions are used all the approaches have a AUC score
of at least 0.76, except BKT with an AUC of 0.54. We can also
observe that the performance of DKT progressively decreases
when the size of the training dataset decreases. This result
is expected as it is well known that deep neural networks
are highly dependent on the size of the training dataset. On
the contrary, the Glicko and GPCF models are not severely
affected by changing the size of the training dataset, as these
approaches can work even without a training dataset.

With the PAL dataset, we evaluate the accuracy of the
model under two different conditions: when the hidden
knowledge level is constant over time and when it varies.
The results show that the accuracy of Glicko and GPCF, while
remaining very high, are slightly affected by the temporal

### Table 1

<table>
<thead>
<tr>
<th>Dataset</th>
<th>AUC 0</th>
<th>AUC 1</th>
<th>AUC 2</th>
<th>AUC 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>DKT</td>
<td>0.85</td>
<td>0.87</td>
<td>0.89</td>
<td>0.91</td>
</tr>
<tr>
<td>KDD</td>
<td>0.82</td>
<td>0.84</td>
<td>0.86</td>
<td>0.88</td>
</tr>
<tr>
<td>PAL</td>
<td>0.83</td>
<td>0.85</td>
<td>0.87</td>
<td>0.89</td>
</tr>
</tbody>
</table>
Table 1: Area Under the Curve (AUC) results for all datasets and approaches tested. (*) Glicko and NOCF do not use a training set, thus their performance remain unchanged in these conditions.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>No. of priors</th>
<th>No. of concepts</th>
<th>No. of interactions</th>
<th>BKT</th>
<th>DKT</th>
<th>Glicko</th>
<th>NOCF</th>
<th>GPCF</th>
<th>Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>DKT DATASET</td>
<td>2000</td>
<td>5</td>
<td>50</td>
<td>0.54</td>
<td>0.76</td>
<td>0.77*</td>
<td>0.79*</td>
<td>0.78</td>
<td>X</td>
</tr>
<tr>
<td>DKT DATASET</td>
<td>100</td>
<td>5</td>
<td>50</td>
<td>X</td>
<td>0.71</td>
<td>0.77**</td>
<td>0.79*</td>
<td>0.79</td>
<td>X</td>
</tr>
<tr>
<td>DKT DATASET</td>
<td>20</td>
<td>5</td>
<td>50</td>
<td>X</td>
<td>0.68</td>
<td>0.77**</td>
<td>0.79*</td>
<td>0.79</td>
<td>X</td>
</tr>
<tr>
<td>DKT DATASET</td>
<td>10</td>
<td>5</td>
<td>50</td>
<td>X</td>
<td>0.60</td>
<td>0.77**</td>
<td>0.79*</td>
<td>0.79</td>
<td>X</td>
</tr>
<tr>
<td>PAL CONSTANT</td>
<td>26</td>
<td>36</td>
<td>30*36 (1080)</td>
<td>X</td>
<td>0.54</td>
<td>0.77</td>
<td>0.76</td>
<td>0.79</td>
<td>0.82</td>
</tr>
<tr>
<td>PAL DYNAMIC</td>
<td>26</td>
<td>36</td>
<td>150*36 (5400)</td>
<td>X</td>
<td>0.60</td>
<td>0.76</td>
<td>0.76</td>
<td>0.77</td>
<td>0.79</td>
</tr>
<tr>
<td>KDD</td>
<td>20</td>
<td>10</td>
<td>2000</td>
<td>X</td>
<td>0.68</td>
<td>0.70</td>
<td>0.69</td>
<td>0.71</td>
<td></td>
</tr>
</tbody>
</table>

Changes (losing 0.01 and 0.02 points of AUC, respectively). It is interesting to note that the Oracle (an ideal model that knows the hidden knowledge level of the users) is similarly affected and has a performance very close to GPCF. DKT still shows the worst performance, even if its score increased by 0.06 points of AUC. We hypothesise that the extended duration of the interactions favours DKT by generating a larger dataset.

On the KDD dataset, the performances of all the compared algorithms are particularly close. We note that the performances of Glicko and GPCF have decreased by 0.06, while the performance of DKT remains in the same range as with the other datasets. However, GPCF still remains the highest performing algorithm across all the tested datasets.

The performance of the NOCF variant of GPCF shows the benefits of the collaborative filtering on the proposed model. We can see that in the best scenario, the collaborative filtering approach enables GPCF to gain 0.02 of AUC. In the worst case, the performance decreases by only 0.01. This situation happens with the DKT dataset when there are too few or too many users in the training dataset. We hypothesise that when the training dataset is too small, the priors that it provides are not informative as they do not contain enough data to be representative of the population. On the other hand, when the dataset is too large, it creates a large search space for the fusion operator and the automatic relevance determination is likely to be affected by local optima. We will see in the next section how the collaborative filtering mainly has an important impact on the convergence speed.

4.4 Convergence speed

4.4.1 Static knowledge level

Fig. 6 shows the evolution of the estimation error according to the number of interactions when the knowledge level of the user does not change over time. The results show that GPCF always performs better than the compared approaches. At the very beginning of the interaction, the difference between Glicko and GPCF is not statistically significant, while being significant when compared against the other approaches. Less than two interactions on average for each topic is enough for GPCF to refine its predictions and become the most accurate approach. We also observe that the Data-only method performs significantly worse than the other approaches at the beginning, while progressively improving as the amount of collected data increases. In this experiment, DKT is seriously penalised by the small number of students in the training dataset.

GPCF shows a significant improvement compared to the NOCF and CF-only variants. In particular, GPCF reaches a prediction error lower than 0.15%, significantly faster than the all other approaches: 4.1 times faster than NOCF, 4.0 times faster than Glicko, and 3.38 times faster than Data-only. The other approaches (DKT, CF-Only and the averaging window) never reached this level of accuracy. These results demonstrate that the proposed model leverages the information from the collaborative filter to increase its convergence speed.

4.4.2 Dynamic knowledge level

Fig. 7 shows the evolution of the estimation error over time. GPCF rapidly makes accurate predictions and tracks changes of the students’ knowledge level over time. In the figure, the prediction error of the proposed approach is constantly lower than or similar to all the compared approaches, with the difference being statistically significant throughout most of the experiment duration (see Fig. 7). There is only one short period where the difference between GPCF and Glicko or NOCF is not statically significant. At the end of the experiment, the predictions made by GPCF are several times more accurate than the compared approaches despite that the users’ knowledge level changes of the over time: the median root mean square error is equal to 5.57% for GPCF, 6.97% for NOCF, 9.31% for Glicko, 11.75% for the averaging window, 13.98% for Data-only, 21.84% for CF-only, and 35.63% for DKT. We can also observe from the results that GPCF, NOCF
and Glicko are the only approaches that reach a prediction error lower than 10%. However, GPCF reaches this quality of prediction 6.0 times faster than Glicko. The performance of NOCF is particularly close to GPCF in this experiment because of the random generation of the temporal variations, which reduces the covariance between the prior students and the “new” ones. This explains why CF-only performs significantly worse than in the previous experiment as it is therefore less likely to have two virtual students with similar success rates. However, it is interesting to note that even in this unfavourable situation, GPCF remains the fastest and most accurate approach.

Two additional points can be observed from this figure: (1) The Data-only approach performs similarly to the previous experiment during the first period (we note it still performs significantly worse than GPCF at the very beginning). However, its performance rapidly deteriorates in the middle of the experiment, which is characterised as the period when the success rates of virtual students are increasing over time. This result demonstrates that a model only based on collected data is unable to keep track of the knowledge level of the student, as some data might become irrelevant after a certain period of time; (2) the results of the averaging window show that this method performs significantly better than the Data-only approach and manages to track the knowledge level of the student over time, while being significantly less accurate than the proposed student model. In general, the performance of the averaging window is very close to the performance of Glicko, while GPCF is significantly more accurate as soon as more than 10 data points have been recorded for each knowledge component.

### 4.5 Automatic generation of abstract knowledge components

This experiment analyses the covariance matrix from the kernel function found by the automatic relevance determination (see section 3.3.2) for each of the 20 students for the training set of KDD dataset. We draw three observations from this experiment.

First, the hyper-parameters independently found for each student are similar for 17 of the 20 students, as shown by the distance matrix between the different hyper-parameters (Fig. 8 B). The distance was computed with the Frobenius norm of the squared difference of the two matrices.

Second, by using an average value (over the different students) for the hyper-parameters (Fig. 8 A), we can observe that the produced covariance matrix forms three distinct clusters among the 10 knowledge components. Fig. 8 C shows the projection of the 10 knowledge components in a latent space formed by the three main Eigen-vectors of the obtained covariance matrix. The ellipses around each marker are defined by the Eigen-values of the corresponding dimensions, which define the scale factor of the kernel function in this dimension.

Finally, if we observe the label of the knowledge components, we can see that the student model manages to extract a meaningful abstract representation of the different knowledge components. The cluster in the top-right of Fig. 8 C contains the following four components: (1) Using small numbers, (2) Using simple numbers, (3) Using difficult numbers, and (4) Using large numbers. Similarly, the cluster in the top-left consists of the following components: (1) Removing coefficient, (2) Removing constant, and (3) Removing positive coefficient. While it is easy for us to see the links between these components given their labels, the proposed student model managed to find these relationships directly from the (binary) responses of the students. Indeed, the labels are unknown to the student model (the knowledge components are simply numbered from 0 to 9). Moreover, it is important to recall that the 10 knowledge components used in this experiment have been selected because they are the 10 most represented components in the dataset. They have not been selected based on the similarity of their labels.

This result demonstrates a very interesting ability of GPCF to determine the main abstract knowledge components for the students. For example, in this experiment, the latent representation created by the student model can be used to define three abstract knowledge components (as the centre of each cluster). Fig. 9 (K-M) shows the evolution of the knowledge level of the students on these three abstract knowledge components. We can observe that the student is not making progress in the usage of numbers (first cluster, Fig. 9 K), while progressing in the manipulation of constants (second cluster, Fig. 9 L) and stagnates on the other knowledge components (Fig. 9 M).
Removing coefficient
Entering a given number
Using simple numbers

This work was supported by the EU Horizon2020 Project PAL under Grant 643783-RIA. We wish to thank the members of the Personal Robotics Laboratory for their assistance.

5 Conclusion
In this paper we introduced GPCF, a student model that relies on a data-driven regression algorithm that is capable of tracking the knowledge level of students without making any assumptions about the shape of their learning curve. Moreover, we coupled this student model with a collaborative filtering approach that leverages the information collected from the previous users of the system and addresses the cold-start problem by allowing the student model to make accurate predictions, even when a limited amount of data is available.

We evaluated the proposed model on three datasets of students interacting with an ITS, either in the context of mathematical courses or diabetes-related knowledge. The results show that the proposed model combines the quickness of collaborative filtering with the accuracy of approaches that average out large amount of data. The results also demonstrate that the model is able to track the changes of the student’s knowledge level, even when it is rapidly varying. In particular, the predictions made by the proposed model at the end of the experiments are several times more accurate than those of the compared approaches, including methods from the state-of-the-art. The experiments also illustrated that the proposed model automatically determines the relationships between knowledge components by observing their mutual influence. This ability enables the creation of abstract knowledge components, which can be used to provide a summary of the current knowledge state of the students. In conclusion, GPCF is a versatile model that can be used in ITS that are designed to accompany students during extended periods of time.

In our future work, we plan to evaluate our model with more students and to use its knowledge level predictions to personalise the selection of upcoming activities, so as to maintain the student’s zone of proximal development and to maximise the benefits of the tutoring sessions. We will also investigate how recent advances in the design of neural networks through neuroevolution can be leveraged in this context to create better student models [62].

Acknowledgments
This work was supported by the EU Horizon2020 Project PAL under Grant 643783-RIA. We wish to thank the members of the Personal Robotics Laboratory for their assistance.

References
Fig. 9. (A-J) Evolution of the success rate of a student on 10 mathematics knowledge components over time. During the overall interaction, the student answered more than 2000 questions. However, only the first 1000 answers are used to train our model (vertical black line). The evolution depicted after the vertical black line corresponds to long-term predictions from GPCF. The reference data displayed in black is obtained with a Gaussian convolution filter, which has access to the data after the black line. (K-M) Evolution of the success rate on the three abstract-knowledge components found by an Eigen-value decomposition.


[28] J. Beck, “Difficulties in inferring student knowledge from observa-


**Antoine Cully** received the M.Sc. and the Ph.D. degrees in robotics and artificial intelligence from the Pierre et Marie Curie University of Paris, France, in 2012 and 2015, respectively, and the engineer degree from the School of Engineering PolytechParis UPMC, in 2012. His Ph.D. dissertation has received several Best-Thesis awards. Between 2015 and 2018, he was postdoc in the Personal Robotics Laboratory at Imperial College London; he is currently a Lecturer at the Department of Computing at Imperial College London. He has published several journal papers in prestigious journals including Nature, IEEE Transactions in Evolutionary Computation, and the International Journal of Robotics Research. His current research interests involve learning algorithms for autonomous robots, artificial cognitive systems, evolutionary robotics, with particular focus on learning methods for autonomous adaptation.

**Yiannis Demiris** (SM03) received the B.Sc. (Hons.) and Ph.D. degrees from the Department of Artificial Intelligence, University of Edinburgh, Edinburgh, U.K. He is a Professor with the Department of Electrical and Electronic Engineering, Imperial College London, London, U.K., where he holds the Royal Academy of Engineering Chair in Emerging Technologies and heads the Personal Robotics Laboratory. His current research interests include human-robot interaction, machine learning, user modeling, and assistive robotics. He has published over 150 journal in the above areas. Prof. Demiris was a recipient of the Rectors Award for Teaching Excellence in 2012 and the FoE Award for Excellence in Engineering Education in 2012. He is a Fellow of IET, BCS, and Royal Statistical Society.