Student Progress Modeling with Skills Deficiency Aware Kalman Filters

Carlotta Schatten and Lars Schmidt-Thieme
Information Systems and Machine Learning Lab,
University of Hildesheim, Universitätsplatz 1, 31141, Hildesheim, Germany

Keywords: Performance Prediction, Kalman Filter, Matrix Factorization, Student Simulator, Sequencing, Progress Modeling.

Abstract: One new usage of Learning Analytics in Intelligent Tutoring Systems (ITS) is sequencing based on performance prediction, which informs sequencers whether a student mastered or not a specific set of skills. Matrix Factorization (MF) performance prediction is particularly appealing because it does not require tagging involved skills in tasks. However, MF's difficult interpretability does not allow to show the student's state evolution, i.e. his/her progress over time. In this paper we present a novel progress modeling technique integrating the most famous control theory state modeler, the Kalman Filter, and Matrix Factorization. Our method, the Skill Deficiency aware Kalman State Estimation for Matrix Factorization, (1) updates at each interaction the student's state outperforming the baseline both in prediction error and in computational requirements allowing faster online interactions; (2) models the individualized progress of the students over time that could be later used to develop novel sequencing policies. Our results are tested on data of a commercial ITS where other state of the art methods were not applicable.

1 INTRODUCTION

In Intelligent Tutoring Systems (ITS), adaptive sequencers can take past student performance into account to select the next task or feedback which best fits the student's learning needs. One way to approach the problem is based on assessing the student skills and matching them to the required skills and difficulties of the available tasks. In this paper we want to go a step forward with respect to domain independent performance prediction. From an approach informing only on the current/next state of the user, we move to progress modeling, where the students' state has to evolve in a meaningful, plausible and therefore interpretable way over time. In this scenario three problems arise:

1. Tagging tasks with required skills necessitates experts and thus is a time-consuming, costly, and, especially for fine-grained skill levels, also potentially subjective.
2. Progress modeling requires the interpretability of performance prediction's models that should instruct with the student's inferred state efficient sequencing policies.
3. Student progress modeling updates need to occur online so that each event can be used as refinement of the prediction.

Problem (1) involves common performance prediction methods and their extensions: Bayesian Knowledge Tracing (BKT) (Corbett and Anderson, 1994) and Performance Factors Analysis (PFA)(Pavlik et al., 2009). Therefore, other algorithms like Matrix Factorization (MF) were proposed, which are domain agnostic and do not require the authoring effort of skills' tagging (Schatten et al., 2015). By using MF and a simple policy inspired by Vygotsky's concept of Zone of Proximal Development (Vygotsky, 1978), the so-called Vygotsky Policy Sequencer (VPS) obtained comparable results with state of the art rule based sequencers without using rich experts' knowledge (Schatten and Schmidt-Thieme, 2014). As pointed out by Manouselis et al. (2011), too high requirements for intelligent components dramatically affect their integration. Therefore, domain independence is particularly appealing because it allows the integration of adaptive components in large ITS that do not possess the skills involved in the tasks and cannot invest the effort of tagging all their contents.
If MF is able to solve Problem (1), it unfortunately suffers from Problem (2), i.e. it is able to predict next performances without domain information, but the parameters of the model cannot be used to interpret the current state of the user and therefore its progress over time. Moreover, MF online update suf-

Schatten, C. and Schmidt-Thieme, L.
Student Progress Modeling with Skills Deficiency Aware Kalman Filters.
In *Proceedings of the 8th International Conference on Computer Supported Education (CSEDU 2016) - Volume 1*, pages 31-42
ISBN: 978-989-758-179-3

fers of Problem (3). When MF is used for Item Recommendation, its main application, the time affects the users differently than in ITS, since voting movies in different permuted orders will not affect the user's ratings. For this reason it is possible to model user evolution and item characteristics in aggregated time slices, where more subsequent ratings are considered as generated from the same static model. On the contrary, in ITS after each exercise the students learn something according to their learning rate. If a student sees exercises in order or in reverse order of difficulty there will be not only a change in the scores obtained, but also in the acquired knowledge. In order to be able to use the latest model for each decision, an online updating model is required. Ideally this should happen in ITS at event level to reflect, in the future, also the influence of feedbacks and hints.

To overcome these three problems, we developed domain independent progress modeling by integrating the MF algorithms with Kalman Filters, one of the most famous state modeling techniques of control theory. This is achieved by exploiting equations of a student simulator, which mimics the learning process of a student.

As a result, the model:

- has reduced computational requirements,
- remains domain independent,
- has a reduced prediction error,
- is less sensitive to the lack of user data,
- and is made interpretable.

The presented paper is organized as follows. We first introduce the state of the art about students' performance prediction done with MF (Sec. 2). Then, in Sec. 3 the algorithms of MF and its online update are explained. After the Kalman Filters are presented to the reader in Sec. 4, we combine the latter with MF and the equations of a student simulator to obtain a student progress modeler. Finally, in the Experiment Section (Sec. 5), we analyze the novel algorithm under different perspectives, which involves prediction error and progress modeling.

2 STATE OF THE ART

MF has many applications like, for instance, dimensionality reduction, clustering and also classification (Cichocki et al., 2009), but its most famous application is for Recommender Systems (Koren et al., 2009), where the algorithm recommends items to a user by predicting the ratings (s)he would give to them. Recently the algorithm was extended to time modeling. In papers predicting movie ratings or doing item recommendations, such as Xiong et al. (2010) and Li et al. (2011) the time is modeled with time slices, so that the user's model needs to be updated at each time slice. Because of that, no forecasting can be done for time slices after the current and last one, for which data are available. More similar to our approach is the online method proposed in Rendle and Schmidt-Thieme (2008) that we explain in detail in Section 3.2. There, for each new sample available the model of the user is accordingly updated. Unfortunately, the algorithm requires for each update the entire student's history. Therefore, computational requirements for systems that work in real time performances, such as ITS, becomes too restrictive. As reported by Schatten et al. (2015) 6 seconds were required for an update, whereas real time performances should stay under the 0.1 seconds threshold (Nielsen, 1994).

As we explained in the introduction, when the aforementioned algorithms are applied in a new domain other problems arise. The first time that MF was applied to ITS, Thai-Nghe et al. (2011, 2010, 2012) associated users with students, items with tasks, and ratings to the probability of a correct answer at first attempt. Alternatively, as in Schatten et al. (2014a) and Voss et al. (2015), ratings could be associated to the percentage of correctly answered questions. Also other Machine Learning techniques have been used to model the students' state. Bayesian Knowledge Tracing (BKT) is built on a given prior knowledge of the students and a data set of binary students' performances. It is assumed that there is a hidden state representing the knowledge of a student and an observed state given by the recorded performances. The model learned is composed by slip, guess, learning and not learning probability, which are then used to compute the predicted performances (Corbett and Anderson, 1994). In the BKT extensions difficulty, multiple skill levels and personalization are taken into account separately (Wang and Heffernan, 2012; Pardos and Heffernan, 2010, 2011; D Baker et al., 2008), whereas in our framework those aspects are considered at the same time. MF most famous advantage in comparison to BKT is the reduced authoring effort, since experts are not requested to insert the required skills to solve a task or use a hint. However, MF computed parameters cannot be associated to the student's knowledge as BKT modeled skills (Pardos and Heffernan, 2011). In this paper we want to develop a progress modeling algorithm based on MF online-updating performance prediction that can work with fast performances to schedule the recommendation of tasks, hints and feedbacks. Usual approaches for sequencing are Reinforcement Learning techniques, which are applicable in ITS with strong restrictions (Schatten et al., 2014b),

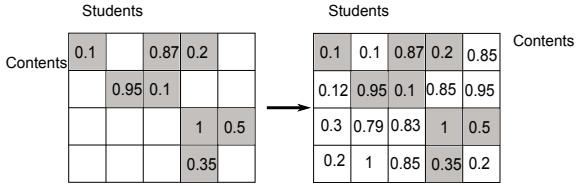

Figure 1: Table of scores given for each student on tasks (or interacting with generic tasks) (left), completed table by the MF algorithm with predicted scores (right).

since the collection of an exploratory corpus implies frustrating users with either too easy and too difficult tasks or random hints and feedbacks. Therefore, we started from the implementation of Schatten et al. (2015), where MF was used successfully combined with a simple policy to schedule tasks with particular attention to the computational requirements defined in Schatten et al. (2014c). An additional reason for choosing Schatten et al. (2015) is the possible extension to multi-modal data analysis as presented in Janning et al. (2014a) and Janning et al. (2014b), or hint sequencing as suggested by Schatten et al. (2014b).

3 ONLINE MATRIX FACTORIZATION

We define our problem as a tuple $(S, C, \hat{y}, \tau)$ where, given a set S of students, $s_i \in \{1, \dots S\}$ is the i–th student modeled as a vector $\varphi^t \in \mathbb{S} := \mathbb{R}^K$, where K is the number of skills involved and $\mathbb{S}$ is the student's space. C is a set of tasks, where $c_j \in \{1, \dots C\}$ is the j–th task, defined with a vector $\psi_j \in \mathbb{C} := \mathbb{R}^K$ representing the K skills required to solve a task defined in the tasks' space $\mathbb{C}$. In this context we want to find a suitable prediction function able to compute the predicted performance $\hat{y}(\varphi_i, \psi_j)$ of a student s_i on a task c_j considering all his past interactions. In order to do so we need also to find $\tau : \mathbb{S} \times \mathbb{C} \to \mathbb{S}$ a function defining the follow-up state φ^{t+1} of a student s_i after interacting with task c_j. We explain hereafter how this is done with pure MF techniques.

3.1 Static Matrix Factorization

Generally, in Recommender Systems MF predicts which are the future user ratings on a specific item based on its previous ratings and the previous ratings of other users (Koren et al., 2009). The concept has been extended to student performance prediction, where a student's next performance, or score is predicted. The matrix $Y \in \mathbb{R}^{n_s \times n_c}$ can be seen as a table of n_c total tasks and n_s students used to train the system, where for some tasks and students performance measures are given. MF decomposes the matrix Y in two other ones $\Psi \in \mathbb{R}^{n_c \times K}$ and $\Phi \in \mathbb{R}^{n_s \times K}$, so that $Y \approx \hat{Y} = \Psi\Phi^T$. Ψ and Φ are matrices of latent features, where each task c_j, and each student s_i, is represented, i.e. modeled, with a vector of K latent features (ψ and φ respectively). Although these latent features cannot be mapped to an exact meaning as done in BKT technology, in Thai-Nghe et al. (2010) those values were associated with the skills involved in the tasks and the skills of the students. The latent features learned with stochastic gradient descent from the given performances allow computing the missing elements of Y for each student i in each task j of a dataset D (Fig. 1) without manually tagging the skills of the domain. For this reason this approach has been called domain independent in Schatten and Schmidt-Thieme (2014). The optimization function of MF is represented by:

$$\min_{\Psi_j, \varphi_i} \sum_{i,j \in D} (y_{ij} - \hat{y}_{ij})^2 + \lambda(\|\Psi\|^2 + \|\Phi\|^2) \qquad (1)$$

where one wants to minimize the regularized Root Mean Squared Error (RMSE) on the set of known scores. The prediction function is represented by:

$$\hat{y}_{ij} = \sum_{k=0}^{K} \varphi_{ik}\psi_{jk}, \qquad (2)$$

3.2 Online Update

Input: $History_i$, λ,Ψ, β, K, $iter_{Max}$
Output: φ^t
$\Psi \sim N(0, \sigma^2)$
$iter_{Max} = History_i.length * iter_{Max}$;
for *iter = 1 to* $iter_{Max}$ **do**
  Select j randomly from $History_i$;
  $err = y - \left(\sum_{k=0}^{K} \varphi_{ik}\psi_{jk}\right)$;
  for *k = 1 to K* **do**
    $\frac{\partial err}{\partial \varphi_{ik}} += \beta\left(err * \psi_{jk} - \lambda\varphi_{ik}\right)$;
    update φ_{ik};
  end
end

Algorithm 1: UpMF Rendle et al (2008), where β is the learning rate, λ is the regularization parameter, Ψ are the tasks' latent features, $iter_{Max}$ is the number of algorithm's iterations, $History_i$ are all the tasks IDs j the student i interacted with with performance y.

One of the criticized problems of MF is that it does not deal with time, i.e. the latent features are constant after the first training. In order to keep the model up to date, Schatten et al. (2015) implemented, in a large commercial ITS, the online update proposed in Rendle and Schmidt-Thieme (2008). The update, that

we will call hereafter UpMF, consists in solving again the minimization problem of Eq. (1) optimizing only φ with stochastic gradient descent algorithm. This means the student's model is learned at each interaction from scratch. Schatten et al. (2015) coherently with Rendle and Schmidt-Thieme (2008) noticed that after approximately 20 interactions the model update's error for UpMF was degenerating. Schatten et al. (2015) overcame the problem by retraining the model each night, assuming students would see approximately 10 tasks per day. This was of course imposing strong requirements on the machine where the application ran since the training is more demanding computationally in comparison to the prediction phase. According to the pseudo-code Alg. 1 reported, there are two main limitations of this algorithm. The first one is the dependency between the history length and the number of algorithm's iterations required to converge to a solution. The more student's interactions are available, the more iterations are needed by UpMF to converge (see Alg. 1). As a consequence the time required to update the model increases over time. To keep the update time constant one should select meaningful samples out of the given history. Unfortunately, we are not aware of previous work analyzing this aspect in detail. The second issue is related to the invariance to the samples sequence, i.e. when a sample is selected out of the ones available old and new ones are considered equally. This means that the sequence has no influence in the model computation.

4 KALMAN STATE ESTIMATION FOR MF (KSEMF)

In this Section we present a novel update method that overcomes the main issues of the current state of the art. Kalman Filters are one of the most used state estimation algorithms in operations research (Kalman, 1960) and therefore constitute a valid approach to our progress modeling problem. First of all the sequentiality of the measurements plays a major role. Then, for their recursive structure they do not require the load of the entire student's history to compute the update, so that the update time is constant. Finally, thanks to our approach, we maintain the domain independence of the baseline.

4.1 Kalman Filter Theory

The state x at time t is modeled as a linear combination of the state at time $t-1$ and a control input u at time $t-1$ with additive Gaussian noise $\mathbf{w}$ (Eq. (3)), where A and B are matrices of coefficients multiplying the state and control variables respectively.

$$x_{t+1} = Ax_t + Bu_t + w_t \tag{3}$$

In Eq. 4 the measurements of the environment are predicted adding the current state estimation multiplied by a coefficient matrix H to Gaussian noise $\mathbf{v}$.

$$y_{t+1} = Hx_t + v_t \tag{4}$$

Instead of learning from scratch the student's parameters after each interaction, the Kalman Filter updates its estimation at each time step with predict (Eq. (5)) and correct (Eq. (6)) phases integrating in its prediction the novel available information. Kalman Filters predict the current state $\hat{x}_t^-$ and the error covariance matrix P_t^- by means of Eq. (5), where Q is the state noise covariance matrix derived from the Gaussian noise variance $\mathbf{w}$ of the state variables.

$$\begin{aligned} \hat{x}_{t+1}^- &= A\hat{x}_t + Bu_t \\ P_{t+1}^- &= AP_tA^T + Q \end{aligned} \tag{5}$$

Then, with a new measurement y_t, state estimation $\hat{x}_t$ and error covariance matrix P_t are corrected with Eqs. 6, where K_t is the so-called Kalman Gain and R the measurement noise covariance matrix derived from the variance of the measurement noise v_t.

$$\begin{aligned} K_t &= P_{t+1}^- H^T \left(HP_{t+1}^- H^T + R\right)^{-1} \\ \hat{x}_{t+1} &= \hat{x}_{t+1}^- + K_t\left(y_t - H\hat{x}_{t+1}^-\right) \\ P_k &= (I - K_tH)\,P_t^- \end{aligned} \tag{6}$$

R, Q and P_0 are all diagonal matrices whose values are treated as hyperparameters, e.g. $Q = diag(0.01)$ means that all Q values on the diagonal are assigned to 0.01. We want to use this approach to model the evolution over time of the MF's latent features and consequently show the students' progress over time.

4.2 Kalman State Estimation for Matrix Factorization (KSEMF)

In this Section we present our novel method for progress modeling: the Kalman State Estimation for Matrix Factorization (KSEMF). In order to integrate the Kalman Filter and MF we first need to identify the state and the control of the system. As aforementioned, at each time step φ^t of student s_i, i.e. the student's MF latent features, needs to be updated to φ^{t+1} with a function τ. Under this interpretation, φ_i^t should be the evolving state. The control over the system are the tasks' latent features ψ_j presented to the student, whereas the score y_t represents the measurement and its prediction $\hat{y}_t$ at time t (Eq. (7)). Since this algorithm is modeling the state and the interaction with the

environment explicitly, a working Kalman Filter does not only show that the approach is valid for performance prediction, but also that (1) the students' latent features can be interpreted as the students' state and that (2) the tasks' latent features can be interpreted as the tasks' characteristics.

$$\begin{bmatrix}\varphi_1\\ \vdots\\ \varphi_k\end{bmatrix}_{t+1} = A\begin{bmatrix}\varphi_1\\ \vdots\\ \varphi_k\end{bmatrix}_t + B\begin{bmatrix}\psi_1\\ \vdots\\ \psi_k\end{bmatrix}_t + \mathbf{w}_t$$
$$\hat{y}_{t+1} = H\begin{bmatrix}\varphi_1\\ \vdots\\ \varphi_k\end{bmatrix}_t + \mathbf{v}_t \quad (7)$$

In order to integrate the prediction function of MF (Eq. (2)) we formalized the relationship between state φ_i^t and predicted measurement $\hat{y}_t$ as in (8), having then $H = \psi^T$.

$$\hat{y}_t = \begin{bmatrix}\psi_1\\ \vdots\\ \psi_k\end{bmatrix}_{t-1}^T \begin{bmatrix}\varphi_1\\ \vdots\\ \varphi_k\end{bmatrix}_{t-1} + \mathbf{v}_{t-1} \quad (8)$$

Still missing is Eq. (3), i.e. the function τ mapping the state ${\varphi_i}^t$ with the state at time $t+1$.

4.3 Skill Deficiency aware KSEMF(KSEMF_SD)

In order to make KSEMF aware of the student's skills deficiency, we will model the update function τ, which represents the learning from one task interaction, in a specific way. We started from the simulated student developed in Schatten and Schmidt-Thieme (2014) that was able to simulate a learning process with continuous knowledge and score representation and tasks with multiple difficulty levels. Nevertheless, we do not exclude the possibility to use also other equations to model the relationship between φ^t and φ^{t+1}. The simulator models a learning process defined by the Zone of Proximal Development (ZPD) (Vygotsky, 1978), i.e. a student can learn from a task only if it is of the correct difficulty level. This is defined in the simulated environment as the difference $\alpha^{i,j}$ between the skills of the student φ_i^t and those required to solve the task ψ_j. As a consequence $\alpha^{i,j}$ represents the skill deficiency of the student.

$$\begin{aligned}\tilde{y}(\varphi_i,\psi_j) &= \max(1-\frac{||\alpha^{i,j}||}{||\varphi_i||},0)\\ \tau(\varphi_i,\psi_j)_k &= \tilde{y}(\varphi_{ik},\psi_{jk})\alpha_k^{i,j}\\ \alpha_k^{i,j} &= \max(\psi_{jk}-\varphi_{ik},0)\end{aligned} \quad (9)$$

In Eq. (9) $\tilde{y}$ represents the simulated score of the student and the skills are positively definite. Therefore $\varphi_{ik} > \varphi_{i2k}$ means student i is more knowledgeable than student $i2$ and $\psi_{jk} > \psi_{j2k}$ means task j is more difficult than task $j2$. Finally $\psi_{jk} < \varphi_{ik}$ means a task j is too easy for student i and (s)he cannot learn from it (Schatten and Schmidt-Thieme, 2014). To develop the Skill Deficiency aware Kalman State Estimation for Matrix Factorization (KSEMF_SD) we interpreted the simulator modeled skills ψ_{jk} and φ_{ik}, for all i, j, and k as the from MF computed latent features. We then reformulated the equations modeling the process, Eq. (9), to fit Eq. (3) and work also with negative latent features. Therefore, we slightly modified Eq. (9) to Eq. (10). These changes allowed also negative latent features, but kept the ZPD properties of the simulator, i.e. a student cannot learn from too easy tasks and learns from a task proportionally to his knowledge and the skills required to solve the task. The equations were changed as shown in Eq. 10.

$$\begin{aligned}\tilde{y}(\varphi_i,\psi_j) &= \max(1-\frac{||\alpha^{i,j}||}{||\varphi_i||},0)\\ \tau(\varphi_i,\psi_j)_k &= \tilde{y}(\varphi_{ik},\psi_{jk})\gamma\delta(\alpha_k^{i,j}>0)\psi_{jk}\\ \alpha_k^{i,j} &= \psi_{jk}\max(1-\varphi_{ik}/\psi_{jk},0),\end{aligned} \quad (10)$$

where γ is a weight and δ is a Kronecker δ that is equal to 1 when its condition $\alpha_k^{i,j} > 0$ is verified and 0 elsewhere. $\alpha_k^{i,j} > 0$ for $\max(\psi_{jk}-\varphi_{ik},0)$ when $\psi_k > 0$ and for $\min(\psi_{jk}-\varphi_{ik},0)$ for $\psi_k < 0$. Under the interpretation $\varphi_{i,k} = 0$ means student i does not possess skill k and $|\varphi_{i,k}| > 0$ now means having some ability in skill k. The mathematical properties of the equations did not change much from the previous version and are:

1. The simulated performance $\tilde{y}$ of a student on a task decreases proportionally to his skill deficiencies w.r.t. the required skills.
2. The student will improve all the required skills of a task proportionally to his simulated performance $\tilde{y}$, his learning rate γ up to the skill level a task requires.
3. As a consequence it is not possible to learn from a content more than γ times the required skills.
4. A further property of this model is that tasks requiring twice the skills level a student has, i.e. $\|\psi_j\| \geq 2\|\varphi_i\|$, are beyond the reach of a student.

Given Eq. (10) we obtained
$\varphi_{ik}^{(t)} = \varphi_{ik}^{(t-1)} + (\tilde{y}(\varphi_{ik},\psi_{jk})\gamma\delta(|\varphi_{ik}| < |\psi_{jk}|))\psi_{jk}$,
i.e.

$$A = diag(1)$$

and

$$B = diag(\tilde{y}(\varphi_{ik},\psi_{jk})\delta(|\varphi_{ik}| < |\psi_{jk}|)\gamma). \quad (11)$$

5 EXPERIMENT SECTION

In this Section we analyze different aspects of the algorithm. First, we describe the dataset used for the experiments; then, we analyze the hyperparameters' selection and the model initialization. Afterwards, we discuss the ability of the algorithm to model the student progress. This is done from different perspectives, which involve the personalization of the state and the update rate. sensitiveness of the algorithm to the lack of data.

5.1 Dataset Characteristics

To test the presented algorithm and model the progress of the students we use the dataset collected with an ITS with 20 topics about maths for children aged from 6 to 14, who can practice with over 2000 tasks at school or at home.
An example of questions proposed to the students can be found in Fig. 2. From these questions proposed in sets, that we call interactions or tasks, we do not know which ones precisely were answered correctly since the ITS aggregates the information in a single score. For these multiple-skills interactions we do not possess the skills involved, therefore, in this context, we cannot use classic BKT and PFA approaches. The score, as in Schatten and Schmidt-Thieme (2014), is represented in a continuous interval which goes from 0 to 1. The topics and new skills to be acquired are introduced following the curriculum of the country. The tasks are presented with a rule-based sequencer, which increases the difficulty of the tasks once the student completed and passed all the tasks of the difficulty level. If the tasks are not passed the student gets a regression exercise or can try again to solve the task.

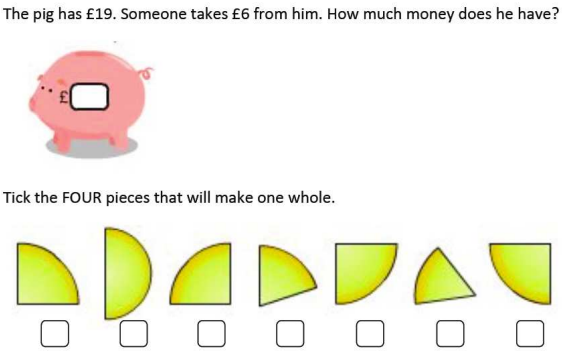

Figure 2: Two questions of the commercial ITS.

Of the large dataset of the commercial ITS we selected two subsets described in Tab. 1 in order to minimize the noise due to the lack of data and monitor the progress of the error and latent features over time. Consequently, we selected only the students i with at least $N_i > 10$ interactions, where one interaction correspond to a student solving a set of 10 questions aggregated in a single score. The available students are then divided in two groups. The group of those with $10 < N_i < 100$ is used to initialize the latent features of all algorithms (D_{Train}, Tab. 1), whereas the others with $N_i > 100$ are used to test online updates UpMF and KSEMF_SD (D_{Test} Tab. 1).

Table 1: Dataset Statistics.

	D_{Train}	D_{Test}
Number of Tasks	2035	2035
Number of Students	24288	713
Total Student-Task Interactions, N	751109	102038

5.2 Hyperparameters' Selection

All the model hyperparameters of MF, UpMF and KSEMF_SD were selected with a full Grid Search, i.e. the influence on the model error of different combinations of hyperparameters is analyzed in a brute force manner. First MF ones were evaluated considering the RSME obtained with a further split of D_{Train}. 66% of D_{Train} was used to train the MF model and its 34% was used to test the model with the different hyperparameters. UpMF and KSEMF_SD best hyperparameters are then selected in the ranges presented in Tab. 2 according to the performances in D_{Test} in particular we used the value $Total\_RMSE$ computed as in Alg. 2 to evaluate the performances of the algorithm.

Input: D_{Train}, D_{Test}, Q, R, P_0
Use D_{Train} and Eq. (1) to obtain $\Phi^{(t=0)}$ and Ψ;
for ***each*** s_i c_j *interactions in* D_{test} N **do**
  $A = diag(1)$, $H = \psi_j{}^T$;
  Compute B using Eq. (11);
  $\hat{y}$=Predict, Eq. (5);
  Correct, Eq. (6);
  Err+ $= (y - \hat{y})^2$;
end
Total_RMSE=$\sqrt{Err/N}$;

Algorithm 2: Experiments' Framework.

UpMF and MF hyperparameters are λ, β, $iter_{Max}$ and K. In addition to these, KSEMF_SD possesses four more hyperparameters: Q, R, γ, and P_0. The empirical approach is to model Q, R, and P_0 as diagonal matrices and test their diagonal values with a logarithmic scale. The selected hyperparameters are reported in Tab. 3.

Table 2: Hyperarameters rages tested for UpMF and KSEMF_SD.

Parameters	Range	Step
Learning Rate β	0.01-0.1	0.01
Latent Features K	2-120	20
Regularization λ	0.01-0.1 0.001-0.01	0.01 0.001
Number of Iterations $Iter_{Max}$	10-200	10
State Noise Cov. Q	0.00001-1	logarithmic
Error Noise Cov. P_0	0.00001-1	logarithmic
Measurement Noise Cov. R	0.00001-1	logarithmic
Weight γ	0.00001-1	logarithmic

Table 3: Selected hyperarameters UpMF and KSEMF_SD.

Parameters	UpMF	KSEMF_SD
Learning Rate β	0.01	0.01
Latent Features K	102	62
Regularization λ	0.01	0.01
Number of Iterations $Iter_{Max}$	100	25
State Noise Cov. Q	-	0.00001
Error Noise Cov. P_0	-	1
Measurement Noise Cov. R	-	0.001
Weight γ	-	0.001

In the future more efficient approaches to hyperparameters' selection could be used as the ones suggested by Wistuba et al. (2015) and Schilling et al. (2015).

5.3 State Variables Initialization

The next question to answer was how to initialize the latent features of UpMF and KSEMF_SD. Since both algorithms are fully personalized they both suffers from the so called cold-start problem, which occurs when no information is available about the students or the tasks. Therefore, a random initialization of the latent features would lead to very bad performances (Voss et al., 2015). Usual approach to solve the problem is to train a model with the classic MF algorithm and use the computed tasks' latent features to initialize KSEMF_SD and UpMF. These are then kept constant while applying Alg. 2 or Alg. 1. Since D_{Train} and D_{Test} have no overlapping students, the D_{Test} students' cold-start problem is solved by including in D_{Train} data of their first interactions with the ITS, so that their latent features can be learned in a full training. The samples necessary to avoid the cold start problem, both for students and tasks, are generally 10. This amount was empirically defined by Pilászy and Tikk (2009). Since these 10 interactions are not always available, we show also results when just one interaction is included in D_{Train}. The by MF computed students' and tasks' latent features are then used to initialize respectively $\Phi^{(t=0)}$ and Ψ of UpMF and KSEMF_SD. The MF results shown in all the subsequent figures are the ones of the MF used to initialize KSEMF_SD, so that it is possible to see the lift obtained by the KSEMF_SD update.

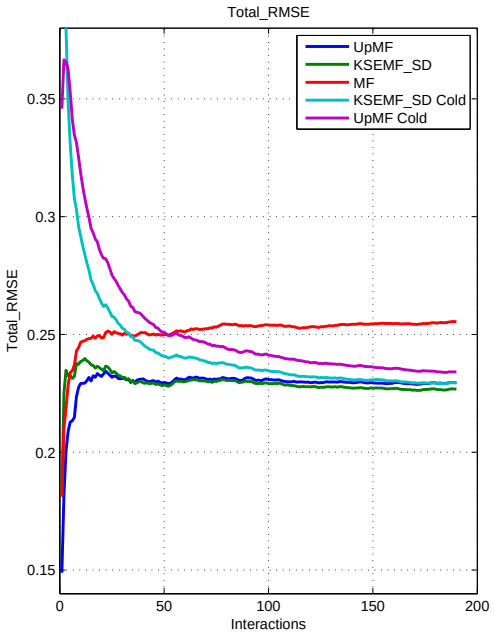

Figure 3: D_{Test} **Total_RMSE behavior over time**: Models marked with "Cold" label are initialized with only 1 interaction in D_{Train} whereas the others with 10.

In Fig. 3 we can see how the Total_RMSE, computed as in Alg. 2, evolves over time. Models marked with the "Cold" label are initialized with only 1 sample whereas the others are initialized with 10. MF Cold behaved like a random predictor with an error around 0.5 and is not shown in Fig. 3. As it is possible to see, the 10 samples substantially improved the error. Nevertheless, we believe this is still not an optimal initialization for KSEMF_SD, since for the first interactions KSEMF_SD is outperformed by UpMF and MF with 10 samples initialization. KSEMF_SD, initialized with 10 samples, has a similar behavior as MF because it inherits the error of MF tasks' latent features whereas KSEMF_SD error amelioration is due to the better students' latent features modeling.

If these 10 interactions are not available, KSEMF_SD Cold converges faster to smaller errors than UpMF Cold. In Voss et al. (2015) it was discussed how the cold start problem limits the usage of MF in small ITS or for short experiments with new students. There-

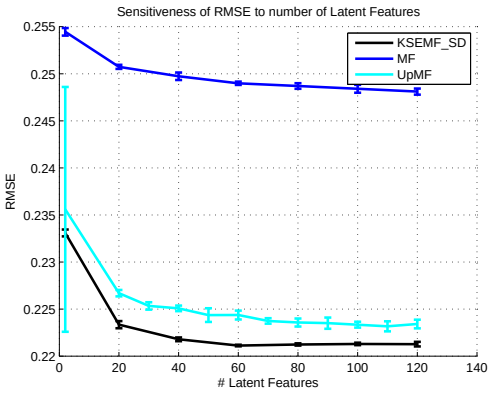

Figure 4: RMSE sensitiveness analysis to latent features.

fore, a faster converging error is an appealing property, that could further reduce the requirements of MF. Despite a better performance in the first interactions UpMF error increases over time. The problem was reported also by Schatten et al. (2015), who, to avoid this issue, retrained the model each night. This is however a quite demanding computational requirement and, as we will see in Sec. 5.5, can affect the progress modeling approach we want to use. Moreover, UpMF requires the entire history of one student as input parameter for Alg. 1 that in case of DB implementation will not only slow down the performances but also increase the complexity of the system. KSEMF_SD does not require demanding DB accesses to extract the entire student's history since it uses only information of the current time step to predict the next one.
Finally, in this experiment we also provide the first proof that KSEMF_SD is able to predict the student performances, meaning that it is possible to interpret the student's latent features as their state.

5.4 RMSE Evaluation

In this Section we evaluate the overall algorithm performances by computing the $Total\_RMSE$, as in Alg. 2. For MF, UpMF and KSEMF_SD we analyze the sensitiveness to the number of latent features. Moreover, we repeated the experiment five times to be able to exclude the variance influence due to the random initialization of the MF. As shown in Fig. 4 the algorithm is able to outperform our reference baselines in all tried latent features configurations.

5.5 Modeling Student Progress

In order to use the developed algorithm to model student progress, it is important to be able to use the performance predictor as model for the user state and take decisions accordingly. One of the claimed disadvantages of MF approaches in comparison to BKT and PFA is that the amount of knowledge of the student cannot be extracted directly from the latent features computed by the algorithms. For this reason Schatten and Schmidt-Thieme (2014) proposed a sequencer which uses only the information coming from the predicted score. In Fig. 5 it is shown (a) how the latent features evolve according to KSEMF_SD algorithm in a scenario with 62 latent features and (b) how the latent features evolve according to UpMF algorithm in a scenario with 102 latent features. Fig. 5 (f) shows the actual score of the student (blue) and the predicted performance of the student by KSEMF_SD (green) and MF (red). In all displayed examples it is not possible to understand what is the overall state evolution of the student. However, the predicting ability of KSEMF_SD let us suppose that the latent feature have indeed a state meaning for the algorithm and consequently an evolution according to the student's performance should be monitored. Therefore, to monitor a meaningful trend, we aggregated the features computing the norm 1 normalized for the number of latent features as in Eq. (12) and depicted the results in Fig. 5 (c) for KSEMF_SD and (d) for UpMF.

$$kn = \frac{1}{K}\sum_{k=0}^{K}|\varphi_k| \tag{12}$$

Under the interpretation that $\varphi_{i,k} = 0$ means student i does not possess skill k, whereas $|\varphi_{i,k}| > 0$ means having some ability in skill k, variable kn could be understood as the personalized knowledge evolution or the learning curve of the user. Although UpMF latent features are learned from scratch after each interaction one can notice in the figures an evolution trend, which is as plausible as the one of KSEMF_SD. This also confirms that the latent features in MF approaches represents the state of the user and their value could be used to retrieve the students knowledge amount. We believe this works because the tasks' latent feature are kept constant. Therefore, in order to keep track of the current state of the students one cannot do a full retrain of the UpMF model, as done by Schatten et al. (2015), since this would reset the values of the tasks' latent features, that allow reconstructing at each interaction the state of the student by means of the student's history.

5.6 Personalization

One important aspect of progress modeling is personalization. MF creates an individualized model as well for tasks as for students. In order to do so also for KSEMF_SD, each student has his/her own

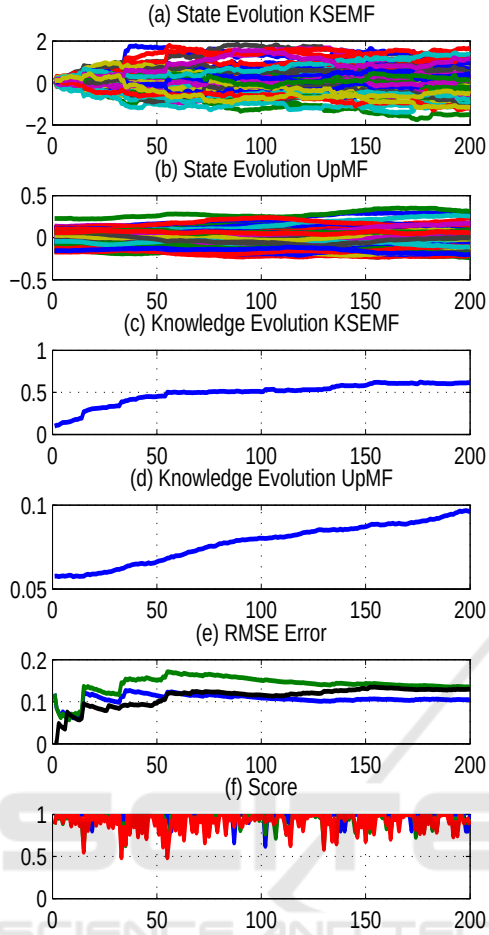

Figure 5: **x-Axis:** Number of tasks seen by the student or interactions. **y-Axis:** *(a)* state evolution according to KSEMF_SD with K=62. *(b)* state evolution according to UpMF with K=102. *(c) and (d):* knowledge evolution for KSEMF_SD and UpMF computed as in Eq. (12). *(e)* Total_RMSE of KSEMF_SD (blue), MF (green) and UpMF (black). *(f)* Actual performance of the student (blue), predicted performance by KSEMF_SD (green), MF (red).

KSEMF_SD equations updating according to his/her modeled state and performances. Since the simulator equations are based on the state variable, in this context, also the B matrix is personalized and change at each interaction. Therefore, the update equations of KSEMF_SD model personalization in two different ways. The B matrix represents the influence of the student on the update, i.e. what is his/her learning rate and its skills' deficiency. The control u, i.e. the tasks' latent features ψ, represents the influence of the task on the knowledge acquisition of the student. Hereafter, we will see how the state as well as the update evolve over time in a personalized way.

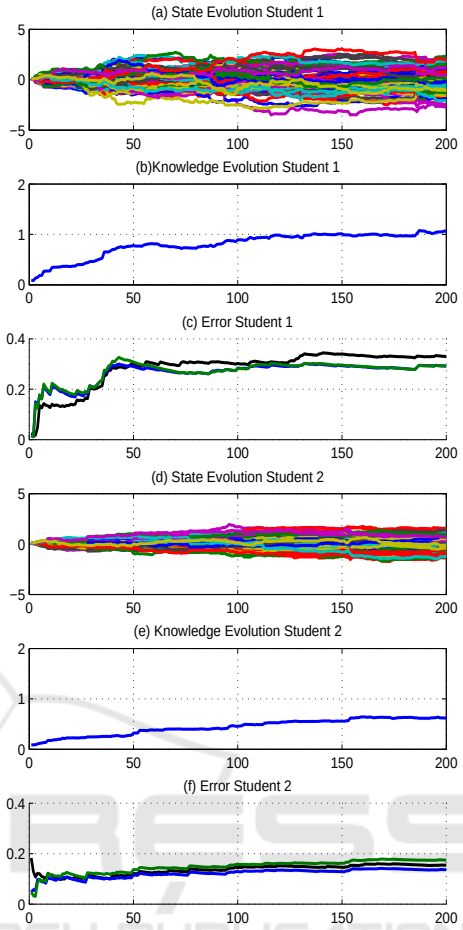

Figure 6: **x-Axis:** Number of tasks seen by the student or interactions. **y-Axis:***(a) and (d):* KSEMF_SD state evolution of two different students, K=62. *(b) and (e): kn* of KSEMF_SD latent features computed as in Eq. 12. *(c) and (f):* Total_RMSE of KSEMF_SD (blue), MF (green) and UpMF (black) of two different students.

5.6.1 Personalized State Evolution

See Fig. 6 (b) and (d) to see the personalized latent features' trends of KSEMF_SD. In Fig. 4 (c) and (f) and in Fig. 5 (e) we can see the Total_RMSEs of the models for three specific students. These are overall coherent with the results presented in Fig. 3. This information could be used in several ways, e.g. by later establishing the mapping between the computed *kn* trend and the actual knowledge acquisition of the users, we could design novel policies for sequencing tasks, feedbacks and hints. In addition, the relationship between *kn* and the model error should be further analyzed. This will allow also to monitor the performances of the performance predictor over time.

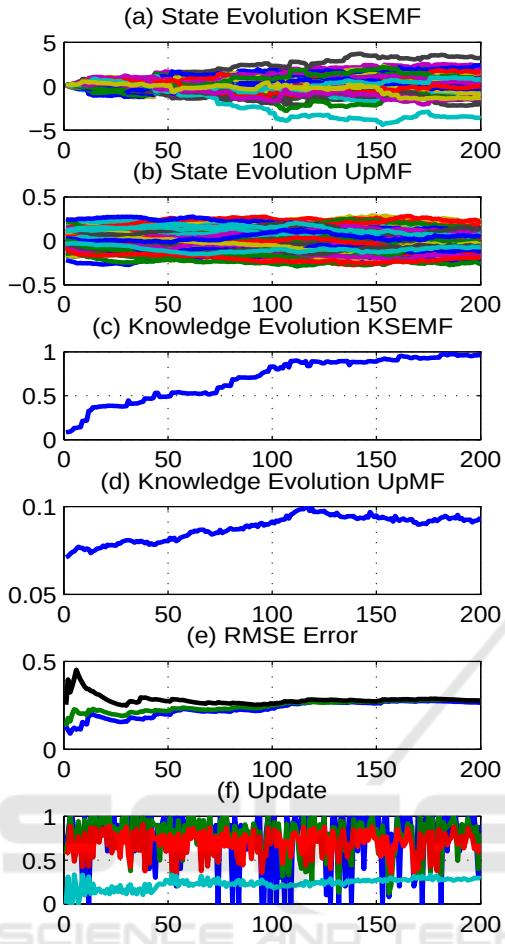

Figure 7: **x-Axis:** Number of tasks seen by the student or interactions. **y-Axis:** *(a)* how the state evolves according to KSEMF_SD with K=62. *(b)* shows how the state evolves according to UpMF algorithm with K=102. *(c)* and *(d)* show the knowledge evolution, computed as in Eq. (12). *(e)* RMSE of KSEMF_SD (blue), RMSE of MF (green) and UpMF (black). *(f)* Actual Performance of the student (blue), predicted performance of the student by the KSEMF_SD (green), predicted performance by MF (red) and $\tilde{y}$ (turquoise).

5.6.2 Personalized Update Evolution

In this Section we discuss the plausibility of the personalized update trend derived through Eqs. (10). For simplicity we considered $\tilde{y}$, which represents the update of the state, since it is later multiplied with constant γ to obtain B (See Eq. 3). In Fig. 7 (f) we show, for a student, how $\tilde{y}$ evolves over time. An almost constant update is plausible, since it mimics the learning rate of the student, which is related to his/her learning ability. However, its adaptive computation through the state is of advantage, since it allows the model to faster adjust to the students' states changes. In Fig. 8

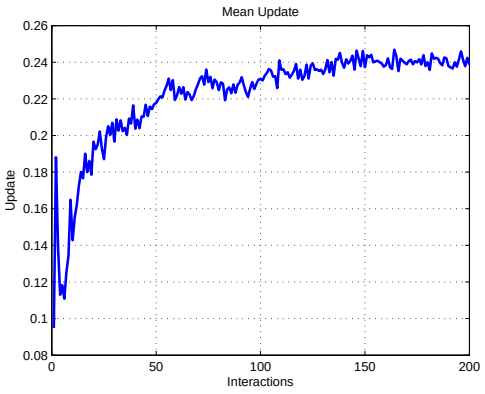

Figure 8: **Mean Update Over Time** Update behavior at each interaction on average for all students.

we see that the average update for all students evolves over time converging only in the last interactions to a constant value. This is explicable with the previously seen behavior of KSEMF_SD in the first interactions (see Fig. 3) and should be seen as another indicator that the initialization of the algorithm is not optimal.
Although we were not interested in keeping the simulation properties of the simulator from which we derived our equations, we briefly discuss why $\hat{y}$ is smaller than the actual performance. Since the variables of φ and ψ are not clipped between 0 and 1 as in Schatten and Schmidt-Thieme (2014) $\left\|\alpha_k^{i,j}\right\|$ is consequently bigger on average and $\tilde{y}$ smaller than the actual performance. In conclusion, given the ameliorated results of KSEMF_SD over UpMF, reported in both Fig. 3 and Fig. 4, we overall showed that the designed equations for KSEMF_SD are suitable to update the students' latent features.

6 CONCLUSIONS

In this paper we presented KSEMF_SD a novel method for student progress modeling based on online updating MF performance prediction and skills' deficiency aware Kalman Filters. We go a step forward with respect to domain independent performance prediction with progress modeling; showing how to represent the evolution of the students over time in a plausible way. This is done by assigning a specific interpretation to latent features which represents the state of the student and the characteristics of a task. In future work, we believe to be able to map the relationship between the computed *kn* and the real knowledge evolution. This will hopefully deliver an effortless analysis tool to teachers and developers.
The developed algorithm also showed appealing properties in comparison to another domain indepen-

dent progress modeler. First, the computational requirements are reduced because the entire student's history is not necessary to compute the updated latent features. Then, the algorithm remains domain independent because the tagged skills of the tasks are not necessary to deliver a score prediction. Finally, KSEMF_SD reduced the prediction error and is less sensitive to the lack of that. In future work we believe to further be able to reduce the error by developing a better initialization of the students' latent features.

ACKNOWLEDGEMENT

This research has been co-funded by the Seventh Framework Programme of the European Commission, through project iTalk2Learn (#318051). www.iTalk2Learn.eu.

REFERENCES

Cichocki, A., Zdunek, R., Phan, A. H., and Amari, S.-i. (2009). *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. Wiley. com.

Corbett, A. and Anderson, J. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *UMAI*.

D Baker, R. S., Corbett, A. T., and Aleven, V. (2008). More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *ITS*, pages 406–415. Springer.

Janning, R., Schatten, C., and Lars, S.-T. (2014a). Feature analysis for affect recognition supporting task sequencing. In *ECTEL*.

Janning, R., Schatten, C., and Schmidt-Thieme, L. (2014b). Multimodal affect recognition for adaptive intelligent tutoring systems. In *FFMI EDM*.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Li, B., Zhu, X., Li, R., Zhang, C., Xue, X., and Wu, X. (2011). Cross-domain collaborative filtering over time. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pages 2293–2298. AAAI Press.

Manouselis, N., Drachsler, H., Vuorikari, R., Hummel, H., and Koper, R. (2011). Recommender systems in technology enhanced learning. In *Recommender systems handbook*, pages 387–415. Springer.

Nielsen, J. (1994). *Usability engineering*. Elsevier.

Pardos, Z. A. and Heffernan, N. T. (2010). Modeling individualization in a bayesian networks implementation of knowledge tracing. In *UMAP*. Springer.

Pardos, Z. A. and Heffernan, N. T. (2011). Kt-idem: introducing item difficulty to the knowledge tracing model. In *UMAP*, pages 243–254. Springer.

Pavlik, P., Cen, H., and Koedinger, K. (2009). Performance factors analysis-a new alternative to knowledge tracing. In *AIED*.

Pilászy, I. and Tikk, D. (2009). Recommending new movies: Even a few ratings are more valuable than metadata. In *RecSys*.

Rendle, S. and Schmidt-Thieme, L. (2008). Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 251–258. ACM.

Schatten, C., Janning, R., and Schmidt-Thieme, L. (2014a). Vygotsky based sequencing without domain information: A matrix factorization approach. In *Computer Supported Education*, pages 35–51. Springer.

Schatten, C., Janning, R., and Schmidt-Thieme, L. (2015). Integration and evaluation of a machine learning sequencer in large commercial its. In *AAAI2015*. Springer.

Schatten, C., Mavrikis, M., Janning, R., and Schmidt-Thieme, L. (2014b). Matrix factorization feasibility for sequencing and adaptive support in its. In *EDM*.

Schatten, C. and Schmidt-Thieme, L. (2014). Adaptive content sequencing without domain information. In *CSEDU*.

Schatten, C., Wistuba, M., Schmidt-Thieme, L., and Gutirrez-Santos, S. (2014c). Minimal invasive integration of learning analytics services in its. In *ICALT*.

Schilling, N., Wistuba, M., Drumond, L., and Schmidt-Thieme, L. (2015). Joint model choice and hyperparameter optimization with factorized multilayer perceptrons. In *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*, pages 72–79. IEEE.

Thai-Nghe, N., Drumond, L., Horvath, T., Krohn-Grimberghe, A., Nanopoulos, A., and Schmidt-Thieme, L. (2011). Factorization techniques for predicting student performance. *Educational Recommender Systems and Technologies: Practices and Challenges. IGI Global*.

Thai-Nghe, N., Drumond, L., Horvath, T., and Schmidt-Thieme, L. (2012). Using factorization machines for student modeling. In *UMAP Workshops*.

Thai-Nghe, N., Drumond, L., Krohn-Grimberghe, A., and Schmidt-Thieme, L. (2010). Recommender system for predicting student performance. *Procedia Computer Science*, 1(2):2811–2819.

Voss, L., Schatten, C., and Schmidt-Thieme, L. (2015). A transfer learning approach for applying matrix factorization to small its datasets. In *EDM2015*.

Vygotsky, L. L. S. (1978). *Mind in society: The development of higher psychological processes*. HUP.

Wang, Y. and Heffernan, N. T. (2012). The student skill model. In *ITS2012*.

Wistuba, M., Schilling, N., and Schmidt-Thieme, L. (2015). Sequential model-free hyperparameter tuning. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 1033–1038. IEEE.

Xiong, L., Chen, X., Huang, T.-K., Schneider, J. G., and Carbonell, J. G. (2010). Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, volume 10, pages 211–222. SIAM.