Parallelizing Automatic Temporal Cognitive Tool for Large-Scale Online Learning Analytics

Tianrui Jiang, Wenjun Wu, and Yanjun Pu
State Key Laboratory of Software Development Environment,
Beihang University, Beijing, China
{jiangtianrui,wwj,puyanjun}@nlsde.buaa.edu.cn

Abstract. With the advent of Massive Online Open Courses (MOOCs), the data scale of student learning behavior has significantly increased. In order to analyze these datasets efficiently and present on-the-fly intelligent tutoring to online learners, it is necessary to improve existing learning analytics tools in a parallel and automatic way. We introduce Automatic Temporal Cognitive (ATC) model to describe temporal progress of online learners and evaluate their mastery of course knowledge. As a complex dynamic Bayesian network model, it often causes high computational overhead of training the ATC model via Probabilistic Programming tools. The time-consuming Monte Carlo sampling adopted by the mainstream implementations renders parameter fitting for the model a slow execution process. To address the issue, this paper proposes to transform the ATC model into the form of nonlinear Kalman filter and presents a new parallel ATC tool based on the Spark framework with the method of Unscented Kalman Filter (UKF). This tool improves the ATC model by using a parallel UKF method with the capability of automatically estimating the parameters in the whole sequential process. Experimental results demonstrate that this tool can achieve the fast execution speed and greatly improve the robustness of training parameters on different sizes of real educational data sets.

Keywords: Learning analytic · Nonlinear state-space model · Kalman filter · Spark framework · Probabilistic programming

1 Introduction

With the development of Internet, online education has greatly increased in both scale and quality. Therefore, more and more students take part in MOOCs courses, which leads to large scale learning behavior data. Learning analytics tools are widely used to estimate the students’ knowledge state through mining their learning behavior data in online educational systems. There are plenty of tools for learning analytics such as Cognitive Diagnosis Model [1,2] and Knowledge Tracing Model [3]. Among them, Automatic Temporal Cognitive Model...
(ATC) [4] can accurately trace a student’s latent knowledge state during his skill acquisition process. Based on the ATC model, developers can build intelligent tutoring services to provide students instant feedbacks or recommendations for their study, and generate assessment reports daily to enable class instructors to quickly evaluate study process. Given the significant increase in the scale of students and dynamics of in-class enrollment, it is important to design an efficient and robust ATC-based intelligent tutoring algorithm for online education systems.

ATC model is a unified and integrated framework that can automatically discover a multiple-dimensional cognitive model and formulate a dynamic learning process over longitudinal student data. This framework enables us to trace the nonlinear dynamic change of multi-dimensional skills including skill improvement and forgetting during a student’s learning process. Moreover, based on it, we can automatically build the cognitive model through student performance data to describe the latent skill vector (Q-matrix [5]) for educational content.

Currently, there are two main problems regarding the ATC parameter tuning and execution efficiency. The first one is how to configure the parameters with their proper probability distribution. Generally, the training result of parameters may have some noise which is probably caused by slip and guessing during the assessment-taking process of students. Because the ATC is essentially a dynamic Bayesian network, it is necessary to adopt Probabilistic Programming [6,7] to infer its parameters. The second problem is how to improve the execution efficiency. The increase in the amount of data often results in longer fitting time for the ATC model. When the data scale reaches tens of thousands of students, the training process can last for several hours or even days. In the field of learning analytics, there are some tools available for speeding up the training process of cognitive learner models, such as Parallelizing Bayesian Knowledge Tracing Tool [8]. Unfortunately, none of them can be applied to the ATC model because their approaches are constrained in the linear state-space framework of Hidden Markov Process, which cannot model nonlinear dynamics in dynamic cognitive skills. As a result, this computation problem for the ATC model hinders the on-the-fly learning analytics in online education platforms that often must handle many learning behavior event records on daily basis.

In this paper, we introduce a Kalman-Filter-based Automatic Parallel (KFAP) tool for parameter estimation of the ATC model. This new tool regards the ATC model as a nonlinear random process and thus uses Singular Value Decomposition-Unscented Kalman Filter (SVD-UKF) to estimate its parameters. Based on the Spark framework, KFAP parallelizes the execution of SVD-UKF computation. The major contributions of KFAP tool in our paper include:

1. Improvement in speed and stability: Using Unscented Kalman Filter to fit parameters of dynamic Bayesian network instead of MCMC (Markov Chain Monte Carlo) sampling, which makes the EM (Expectation Maximization) algorithm applicable. Thanks to it, ATC model has a huge acceleration.
2. Improvement in scale of data: Parallelizing the implementation of SVD-UKF for ATC based on the Spark framework, which makes big data analysis possible.
2 Related Work

2.1 Automatic Temporal Cognitive Model

Automatic Temporal Cognitive Model (ATC) \[4\] is a new cognitive learner model by integrating essential psychometric components of Cognitive Diagnosis Model (CDM) and Knowledge Tracing (KT) Model.

CDM is a kind of cognitive diagnosis techniques that aims to predict student performance by discovering student states from the response of their exercises. It contains Item Response Theory (IRT) \[9\], Deterministic Inputs, Noisy-and gate model, and so on. IRT describes the basic relation between the probability that a student can correctly answer an exercise and his mastery of relevant knowledge. It provides a logistic function to model the probability of getting a correct answer using the parameters including proficiency, difficulty, and discrimination, which is defined as formula:

\[
p_{sq} = f(\alpha_q (\theta_s - \beta_q))
\]

where \(\alpha_q\) means the question discrimination, \(\beta_q\) is the question difficulty, \(\theta_s\) is the student proficiency and \(f\) denotes the sigmoidal function mapping the calculation result of the student skill and question parameters to 0 and 1.

KT is proposed by AT Corbett and JR Anderson \[3\]. It detects an individualized sequence of exercises to the student based on the probability estimates. Based on this approach, they introduced Bayesian Knowledge Tracing (BKT) using the hidden Markov chain. The classic BKT framework can only describe temporal learning process centered around single-dimensional knowledge concepts.

To overcome the limitation of CDM and BKT, we propose ATC model as a general dynamic Bayesian network in \[4\]. ATC is a unified and integrated framework to automatically discover a multiple-dimensional cognitive model and formulate a student model over longitudinal student data. This framework enables us to trace the dynamic change of multi-dimensional skills including skill improvement and forgetting for the student learning process. Moreover, based on the framework, we can automatically build the cognitive model through student performance data to describe the latent skill vector for educational content. Although the complexity of the ATC model ensures better performance in analyzing student learning process, it brings a new challenge in the aspect of model training. The nonlinearity in the ATC model makes it impossible to adopt Expectant Maximization in parameter estimation in the same way as Hidden Markov Model based BKT. Instead, it must rely upon the slow MCMC sampling of probabilistic programming for dynamic Bayesian network. As a result, Stan \[10\] is used to train the ATC model by sampling different combination of parameters and selecting the best optimal result.

2.2 Parallel Algorithm in Model Optimization

Expectation-Maximization (EM) algorithm is an iterative algorithm used for maximum likelihood estimation containing hidden variables. Many parallel algorithms for EM computation have been proposed, such as a generic parallel
implementation of the EM algorithm for computer vision in parallel distributed memory environments [11] and the fast parallel implementation of EM on NVIDIA GPUs using CUDA [12]. Based on the work of Pedro, Cui [8] applied the parallel algorithm in three distributed frameworks including Graph Lab [13], Piccolo and Spark [14]. Hunter implemented a large-scale online Expectation Maximization with Spark Streaming for low-latency applications [15, 16]. Davier et.al introduced the parallel EM algorithm without improvement for Generalized Latent Variable Models and evaluated the overall gain in different CPU environments [17]. In the field of Probabilistic Programming Language (PPL), Masegosa et al. implemented a toolbox for scalable probabilistic machine learning with a special focus on massive streaming data [18]. Within PPL, most research efforts mainly focus on the majorization design during the sampling process. However, the way of sampling has an inborn property of instability that its performance heavily depends on the initial point. There is no clear standard to guarantee the convergence of sampling results delivered by the algorithms.

Among knowledge-related models, DJ Cook, LB Holder, G Galal, R Maglothin proposed an algorithm in parallel graph-based knowledge discovery [19]. Robert J. Hilderman suggested the method of parallel knowledge discovery using domain generalization graphs [20]. Unfortunately, to the best of our knowledge, none of the above publications are suitable for ATC. These researchers didn’t present good solution to the problem of optimizing parameters in ATC, which is a nonlinear probabilistic programming model, with large number of data sets.

2.3 Kalman Filter and Its Extension for Nonlinear Dynamic Systems

Kalman Filter (KF) is a general algorithm for state-space models, which is able to infer latent variables by using a series of observations and estimating a joint probability distribution over the variables for each timeframe. KF can cope with the circumstance when the observation sequence contains statistical noises and other inaccuracies. The typical KF assumes the Gaussian distribution and linear system in target models. When the transmission function becomes nonlinear, modified KF solutions such as Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), Particle Filter (PF) and Deep KF method [21], have to be applied for nonlinear transformation.

UKF uses a series of samples to approximate the probability density function. It takes several key points to express one state and gives a weight to every point. During the calculation, it transforms these key points and determines the new distribution with them. UKF has a stability issue where a covariance matrix is non-positive definite matrix. Influenced by conditions and model disturbances, when the variance matrix loses the property of positive definition, the Cholesky decomposition algorithm in the traditional UT transform cannot sample the Sigma point, resulting in program interruption and poor stability. To solve this problem, a method named Singular Value Decomposition (SVD) can be integrated into the UKF as SVD-UKF [22]. The new algorithm ensures the positive property of the variance matrix via SVD and deliver more robust results.
3 Kalman-Filter-Based Automatic Parallel Tool for ATC Model

The ATC model is a nonlinear state-space model that consists of two main parts: skill embedding for item response and temporal change in skill level.

(1) Skill embedding for item response: Given the ability of student \( s \) and the difficulty of an exercise \( i \), the model can decide the probability \( p_{s,i} \) that present the probability of \( s \) has a correct response on \( i \). We combine latent skill embedding for personalized lesson sequence recommendation [23] and IRT [24] to construct its item response function as formula (2, 3):

\[
q_{s,i} = \frac{\hat{\theta}_s \cdot \vec{a}_i}{\|\vec{a}_i\|} - \|\vec{a}_i\| \tag{2}
\]

\[
p_{s,i} = Pr\left(R_{s,i} = 1 \mid \hat{\theta}_s, \vec{a}_i\right) = \phi\left(q_{s,i}\right) \tag{3}
\]

\( \phi \) is the logistic function convert the possibility between 0 and 1. \( \theta_s \) is the vector which represents the ability of each skill of student \( s \). Vector \( a_i \) represents the required skill level of exercise \( i \), \( R_{s,i} \) is the result of the response of \( s \) on \( i \).

(2) Temporal change in skill level: Given the ability \( \theta_{s,t} \) of student \( s \) at time \( t \) and the improvement \( l_i \) that the exercise \( i \) can offer, we can calculate the ability \( \theta_{s,t+1} \) of student \( s \) at time \( t+1 \), which can be expressed as formula (4, 5):

\[
\theta_{s,(t+1),n} \sim N\left(\mu_{s(t+1),n}, \sigma^2\right) \tag{4}
\]

\[
\mu_{s(t+1),n} = (\theta_{s,t,n} + l_{i,n} \ast \phi\left(q_{s,i}\right)) \ast f(\Delta t) \tag{5}
\]

where \( \theta_{s,t,n} \) means the able of the \( n \)-th skill in the dimension of \( \theta_{s,t} \). \( l_{i,n} \) means the value of the \( n \)-th dimension of the vector \( l_i \). The forget coefficient \( f(\Delta t) \) is relevant to \( \Delta t \) where \( \Delta t \) is the interval between timestep \( t \) and timestep \( t+1 \). Because the transmission function (Eq. 4) represents a Wiener process, KF can be used to estimate the parameters of the ATC model. In order to accelerate the calculation of KF, many methods have been proposed, such as Parallelized sigma-point KF [25] and decentralized structures for parallel KF [26]. However, due to the fact that KF is a matrix-based operation, these methods mainly focus on the acceleration of matrix optimization, which gives little benefit to the improvement of the overall efficiency of model training. In our case, the improvement with pure parallelization in the matrix operation is much less significant compared with the effect of KF parallelization over the large scale of students.

Since the ATC model contains a non-linear transition function, UKF should be applied to process the process of parameter estimation. We choose SVD-UKF to avoid the non-positive definite matrix. Given the inherent independent learning behavior of online students, we decide to build Kalman-Filter-based Automatic Parallel (KFAP) tool containing the method SVD-UKF based on the Spark framework. With the help of it, EM algorithm can be used instead of slow MCMC sampling in STAN. The KFAP structure is illustrated as Fig.1.
The total training process in the KFAP tool is divided into four stages, including sequence extraction, initialization and distribution of data, calculation with SVD-UUKF, collection of results and EM calculation. The original learning sequence data comes from the online data collection system regularly. The original sequence usually contains useless information and needs to be cleaned. The detailed steps are shown as following:

**Step1:** Extract the answer sequence from the original sequence data. Then, divide the data by student ID. In each student’s answer sequence, sort it by chronological order. Finally, Step 1 provides plenty of answer sequences and each sequence represents the time series data of answers of one student.

**Step2:** After getting the time series data, random values are assigned to the initial capacity of learners and the coefficient of difficulty of assessments. Spark can be used and answer sequences will be distributed to computational nodes. Each node receives only a small part of sequence set. Thus, there is no need to worry about how large the scale of online data is because they are divided into little portion marked with learner.

**Step3:** With the initial capacities of learners and the coefficient of difficulty of assessments, the model can use the recursion formula that depicts the transformation of the capacity of learner with his answer sequence. KF needs two parts to calculate: the formula of transition and the formula of emission. After some transformation of ATC, here comes the formula that is suitable for SVD-UUKF:

\[
\theta_t = g(\theta_{t-1}) + \eta, \eta \sim \mathcal{N}(0, Q)
\]

\[
y_t = h(\theta_t) + \epsilon, \epsilon \sim \mathcal{N}(0, R)
\]

\[
g(\theta_t) = (\theta_t + l_t) \cdot f_t
\]

\[
h(\theta_t) = \Phi\left(\frac{\theta_t \cdot a_i^T}{\|a_i\|} - \|a_i\|\right)
\]
Among formula (6, 7, 8, 9), \( \phi \) is logistic function. Vector \( \theta_t \in \mathbb{R}^k \) represents the ability of each skill of student \( s \) at the timestep \( t \). Vector \( a_i \in \mathbb{R}^k \) represents the required skill level of exercise \( i \). The value \( y_t \) is the predicted value at the timestep \( t \). The coefficient \( f_t \) is the effect on forgetting of student \( s \) from timestep \( t \) to timestep \( t + 1 \). The model uses SVD-UKF [22] in this step for the nonlinearity of transition and the robust process. With its help, the model can trace the transformation data of each learner during his whole learning process. Finally, KFAP gives the cross-entropy value of each answer sequence. 

**Step4:** After step3, this model accumulates all cross-entropy and regard this sum as the loss function. With the help of this function, the Expectation Maximization Algorithm (EM), which was introduced by T.K. Moon [27] for more than 20 years is applied. The final procedure is to estimate the model parameters by maximizing the following objective function:

\[
L(\omega) = \text{LogLikelihood} = \sum_S \sum_{L_s} \log P(R | \theta_0, D, G, Q, R) \tag{10}
\]

4 Experiments

4.1 Datasets

The experiments run two real datasets to test KFAP. OLI Biology dataset\(^1\) was collected from Open Learning Initiative online course of biology from 2012 to 2014 include 5186 learners and 4831 unique assessments, with different learning modules such as Lipids, Meiosis, Proteins and so on. Our experiment datasets take each module as an independent input. Students’ correct rates of responses range from 70% to 95%. Tsinghua University Web Learning dataset is acquired from Tsinghua University MOOC platform. It contains both global and university open learning classes. There are many types of data, including classroom lectures, classroom assessments, interaction with multimedia, and after-school assessments. So far, a total of millions of pieces of learning behavior data have been collected. In the experiment, its data has been divided into different courses.

4.2 Improvement During Fitting Process

In the experiment, the model can get the estimation of a student whether he or she can give a correct response to certain assessment. In Fig. 2, one can observe AUC\(^2\) and Log-Likelihood (loglike) at each iteration of gradient descent. Figure 2 shows that after about 10 times of iterations, the rising trend of the loglike curve slows down. Then, the AUC curve keeps up with the loglike curve and stabilizes at about 0.82, which represents a favorable performance of estimation.

---

\(^1\) https://pslcdatashop.web.cmu.edu/Project?id=115.  
\(^2\) Area Under Curve: a value between 0 and 1 that measures the discriminative ability of a binary classifier.
4.3 Execution Time and Robustness

Stan is a state-of-the-art tool for statistical modeling and high-performance statistical computation. Because of the stochastic nature of sampling, Stan is unstable and its result has plenty of randomness. Application of Kalman Filter can effectively reduce the execution time consumed during the period of optimizing. Figure 3 compares the AUC performance between the sampling in Stan and KFAP for ATC model training. In order to get AUC values during different execution moments, we set the break points between several gradient descent processes and export AUC values in KFAP. In the case of running Stan, we repeat the experiments for multiple times with different iteration sampling durations. The increase in sampling time will extend execution time. With the testing data, AUC acquired by KFAP quickly achieves a satisfactory performance and becomes stable after 2–3 h. In contrast, Stan performs in a very unstable way with the initial iterations. One can see that the AUC of Stan sampling ranges from 0.1 to 0.7 during 10000–20000 s. With such a high uncertainty, it takes much longer time for Stan to prepare and optimize its performance. In Zone I of 3, before Stan has run sufficient rounds of sampling, its performance heavily depends on the initial sampling point, which has more uncertainty than Zone II. When the sampling points are accumulated in Zone II, AUC tends to reach a better status than Zone I and generate a probability to express a rough estimation of the optimal parameters. As a contrast, with the KF and gradient descent, KFAP can quickly converge to the local optimal result since the first few of iterations.

4.4 Acceleration with Spark Framework

Spark can distribute a big volume of operations to many executors. Figure 4 shows the variation tendency of execution time with increasing number of executors. In this experiment, a dataset (9k rows) of 445 students is chosen and execution time is sampled by 10 iterations during the fitting process. The curve shows clearly that KFAP can significantly reduce the execution time with the increase in the number of executors. Figure 5 displays the time consumption with different sizes of datasets (with 4k, 8k, 12k and 16k rows) in different number of
executors (iteration time = 5). KFAP keeps achieving a good performance when the scale of datasets become larger. As each executor can work more efficiently with the increase in datasets, KFAP is suitable to be extended to large-scale datasets.

Fig. 4. Execution time with different clusters  
Fig. 5. Execution time with different datasets

Another comparison is made between the efficiency of the model training running on a single machine and on a Spark framework. The single machine contains a CPU of Intel E3-1246, with 8 GB allocated memory. The Spark cluster has 10 machines, 71 cores (with 1 core of master), 372 GB memory (with 20 GB of master). Because of sufficient memory in Spark framework, the algorithm can design more intermediate variables in order to reduce the execution time. The detailed parameters are shows in the Table 1:

| Table 1. The specification of computing node of two methods |
|-------------------------------|---------------|----------------|---------|--------------|----------------|
| Method                      | CPU           | Process speed | Cores  | Processor | Memory        |
| Single machine              | E3-1246 v3    | 3.5 GHz       | 1      | 1           | 8 GB          |
| Spark framework             | Spark with E3-1246 v3 | 3.5 GHz   | 71     | 71          | 372 GB        |

We trained the ATC model with the datasets both on the single machine and parallel implementation with Spark. Figure 6 shows the improvement of execution time with/without Spark boost. In fact, the model with the Spark framework can utilize more CPU cores and larger memory, thus resulting in significant improvement in terms of running speed especially with the large-scale datasets.

Table 2 chooses 4 different modules (A: Mitosis, B: Carbohydrates, C: Meiosis, D: Lipids) from OLI Biology and shows the final result and its execution time of KFAP, indicating that the KFAP tool can ensure the excellent AUC performance for the ATC model. With the increase in the scale of learning datasets increases, we can correspondingly add more computing cores in the Spark framework to speed up the model training process.
5 Conclusion

In this paper, we propose a Kalman-Filter-based Automatic Parallel (KFAP) tool for the ATC, which is implemented on the Spark framework and adopts Singular Value Decomposition-Unscented Kalman Filter. The two major improvements of KFAP includes: (1) Using Unscented Kalman Filter to substitute for Stan, which makes the EM algorithm to fit parameters possible. (2) Paralleling implementation of SVD-Unscented Kalman Filter tool for the ATC model based on the Spark framework. Experimental results confirm that our system achieves performance improvements in both execution time and the robustness of the model training process. Our future work includes generalization of the KFAP to other nonlinear state-space models and investigation into deep Kalman filters.

Acknowledgements. This work was supported by Development Program of China (Funding No. 2018YFB1004502), NSFC (Grant No. 61532004) and State Key Laboratory of Software Development Environment (Funding No. SKLSDE-2020ZX-01). This work is partially funded by Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing, 100191.

References

1. De La Torre, J.: A cognitive diagnosis model for cognitively based multiple-choice options. Appl. Psychol. Meas. 33(3), 163–183 (2009)
2. Hartz, S.M.C.: A Bayesian framework for the unified model for assessing cognitive abilities: Blending theory with practicality. PhD thesis, ProQuest Information and Learning (2002)
3. Corbett, A.T., Anderson, J.R.: Knowledge tracing: modeling the acquisition of procedural knowledge. User Model. User-adapted Interaction 4(4), 253–278 (1994). https://doi.org/10.1007/BF01099821
4. Pu, Y., Wu, W., Jiang, T.: Ate framework: a fully automatic cognitive tracing model for student and educational contents. In: EDM (2019)
5. Barnes, T.: The q-matrix method: mining student response data for knowledge. In: American Association for Artificial Intelligence 2005 Educational Data Mining Workshop, pp. 1–8, AAAI Press, Pittsburgh (2005)
6. Prékopa, A.: Probabilistic programming. Handb. Oper. Res. Manag. Sci. 10, 267–351 (2003)
7. van de Meent, J.-W., Paige, B., Yang, H., Wood, F.: An introduction to probabilistic programming. arXiv preprint arXiv:1809.10756 (2018)
8. Pu, Y., Wu, W., Han, Y., Chen, D.: Parallelizing Bayesian knowledge tracing tool for large-scale online learning analytics. In: 2018 IEEE International Conference on Big Data (Big Data), pp. 3245–3254. IEEE (2018)
9. Embretson, S.E., Reise, S.P.: Item response theory. Psychology Press, London (2013)
10. Montes-Restrepo, V., Strobbe, G., van Mierlo, P., Vandenberghe, S.: Effects of conductivity perturbations of the tri-layered skull on EEG source analysis. J. Clim. 26(22), 8895–8915 (2013)
11. López-de Teruel, P.E., García, J.M., Acacio, M.E.: The parallel EM algorithm and its applications in computer vision. In: PDPTA, pp. 571–578 (1999)
12. Kumar, N.S.L.P., Satoor, S., Buck, I.: Fast parallel expectation maximization for Gaussian mixture models on GPUs using CUDA. In: 2009 11th IEEE International Conference on High Performance Computing and Communications, pp. 103–109. IEEE (2009)
13. Gonzalez, J.E., Low, Y., Gu, H., Bickson, D., Guestrin, C.: Powergraph: distributed graph-parallel computation on natural graphs. In: Presented as part of the 10th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 12), pp. 17–30 (2012)
14. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I., et al.: Spark: cluster computing with working sets. HotCloud, 10(10-10), 95 (2010)
15. Zaharia, T.H.T.D.M., Bayen, A., Abbeel, P., Hunter, T.: Large-scale online expectation maximization with spark streaming. In: EECS, Berkeley.edu, pp. 1–5 (2012)
16. Zaharia, M., et al.: Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12), pp. 15–28 (2012)
17. von Davier, M.: High-performance psychometrics: the parallel-e parallel-m algorithm for generalized latent variable models. ETS Res. Rep. Ser. 2016(2), 1–11 (2016)
18. Masegosa, A.R., et al.: Amidst: a java toolbox for scalable probabilistic machine learning. arXiv preprint arXiv:1704.01427 (2017)
19. Cook, D.J., Holder, L.B., Galal, G., Maglothin, R.: Approaches to parallel graph-based knowledge discovery. J. Parallel Distrib. Comput. 61(3), 427–446 (2001)
20. Hilderman, R.J., Hamilton, H.J., Kowalchuk, R.J., Cercec, N.: Parallel knowledge discovery using domain generalization graphs. In: Komorowski, J., Zytkowski, J. (eds.) PKDD 1997. LNCS, vol. 1263, pp. 25–35. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63223-9_103
21. Krishnan, R.G., Shalit, U., Sontag, D.: Deep kalman filters. arXiv preprint arXiv:1511.05121 (2015)
22. Ma, Y., Wang, Z., Zhao, X., Han, J., He, Y.: A UKF algorithm based on the singular value decomposition of state covariance. In: 2010 8th World Congress on Intelligent Control and Automation, pp. 5830–5835. IEEE (2010)
23. Reddy, S., Labutov, I., Joachims, T.: Latent skill embedding for personalized lesson sequence recommendation. arXiv preprint arXiv:1602.07029 (2016)
24. Sheng, Y., et al.: Markov chain Monte Carlo estimation of normal ogive IRT models in matlab. J. Stat. Softw. 25(8), 1–15 (2008)
25. Azam, S.E., Ghisi, A., Mariani, S.: Parallelized sigma-point kalman filtering for structural dynamics. Comput. Struct. 92, 193–205 (2012)
26. Hashemipour, H.R., Roy, S., Laub, A.J.: Decentralized structures for parallel kalman filtering. IEEE Trans. Autom. Control 33(1), 88–94 (1988)
27. Moon, T.K.: The expectation-maximization algorithm. IEEE Sig. Process. Mag. 13(6), 47–60 (1996)