The Predictive Performance Equation in a Generalized Knowledge Tracing Machine

Florian Sense (florian.sense@infinitetactics.com)
InfiniteTactics, LLC
Dayton, OH, USA

Michael Collins (michael.collins.74.ctr@us.af.mil)
ORISE at AFRL
Dayton, OH, USA

Jong W. Kim (jong.kim@caemilusa.com)
CAE, Inc.
Wright Patterson Air Force Base, Ohio

Michael Krusmark (michael.krusmark.ctr@us.af.mil)
CAE, Inc.
Wright Patterson Air Force Base, Ohio

Tiffany Jastrzembski (tiffany.jastrzembski@us.af.mil)
Air Force Research Laboratory
Dayton, OH, USA

Abstract

Knowledge tracing models are at the heart of the educational data miner's toolkit. Recent efforts have highlighted the commonality of various approaches and proposed overarching modeling frameworks that provide a broadly applicable set of models that can be tailored to specific use cases. Specifically, these are knowledge tracing machines (KTM; Vie and Kashima) and generalized knowledge tracing (GKT; Pavlik, Eglington, and Harrell-Williams). Here, we hope to achieve three goals: First, we point out the similarities between the approaches of KTMs and GKT. Second, we highlight that both frameworks focus on *feature engineering*. We suggest that extending the scope to also include the *predictive engine* creates an even more general and versatile approach—which we tentatively call Generalized Knowledge Tracing Machines. This paper showcases a simple use case for GKTM in which we focus primarily on its distinguishing dimension: the choice of predictive engines. Third, and finally, we demonstrate how an established cognitive model—the Predictive Performance Equation—can be cast as a GKTM. This example further illustrates the utility of the GKTM framework more generally and points to new modeling avenues involving the PPE specifically.

Keywords: Knowledge tracing; feature engineering; knowledge tracing machine; generalized knowledge tracing; predictive performance equation.

Introduction

Modeling the acquisition and retention of knowledge for individual learners is a core concern of researchers in the cognitive modeling community (Pelánek, 2017). Knowledge tracing models are collectively used to this end—both for individual and groups of learners; both descriptively and prescriptively. These models' insights and capabilities are the bedrock of many cognitive/intelligent tutoring applications (Corbett, Koedinger, & Hadley, 2001).

Recent efforts have presented overarching frameworks for knowledge tracing models: Generalized knowledge tracing (Pavlik, Eglington, & Harrell-Williams, 2021) makes it easy to build many well-established models and implement novel models; knowledge tracing machines (Vie & Kashima, 2019) provide a useful framework for combing time-tested modeling techniques with additional input features. It should be noted that these two frameworks are synergistic and share an emphasis on the feature engineering dimension of model building.

We would like to broaden the fruitful discussion kicked off by these two frameworks by posing that besides considerations regarding feature engineering, there is another dimension worth appraising: the choice of the statistical model that learns the relationship between the engineered features and the outcome measure of interest. We will term this dimension the *predictive engine* and believe that it should be considered in frameworks such as GKT and LKT.

These considerations tie in with a larger on-going debate between machine learning and cognitive modeling approaches to student modeling (Sense, Jastrzembski, Mozer, Krusmark, & van Rijn, 2019). In an attempt to benefit from the best of both worlds, we present our considerations along with an attempt to reformulate a well-established cognitive model—the Predictive Performance Equation—such that it matches the GKT/KTM framework. The main advantage of this reformulation is not in the feature engineering step of the model building but in the possibility to easily use different predictive engines for the same set of engineered features. This, we hope, preserves the key cognitive insights hard-coded into the PPE but makes it "machine learning-ready". The results presented here focus on this effort and show how this approach makes the PPE more easily applicable in settings it was not designed for while highlighting the utility

of deliberately considering the predictive engine used to fit a model more generally. To underscore our suggestion's reliance on the earlier work, we tentatively refer to the framework as generalized knowledge tracing machines (GKTM).

The GKTM Framework

As a first step, building a predictive model requires **feature engineering**. Recent work by Pavlik et al. (2021) and Vie and Kashima (2019) are great examples that focus on creating and comparing models that differ in the features that have been engineered. The desired end result of the feature engineering processes is a feature vector that contains all relevant information that we want our model to leverage.

Using this feature vector, the second step is to "build a prediction model, or *learner*, which will enable us to predict the outcome for new unseen objects." (Hastie, Tibshirani, & Friedman, 2009, p. 2) We will refer to this *learner*—the statistical model of choice—as the **predictive engine** (Kuhn & Wickham, 2020). Model fitting is the process of letting the predictive engine learn how the (engineered) features relate to the outcome measure. The work cited above, for example, uses logistic (Pavlik et al., 2021) and probit regression (Vie & Kashima, 2019) as the predictive engine. We want to emphasize that a wide range of predictive engines can be applied to the same feature vector: A neural network, a random forest, or a gradient-boosted decision tree ensemble could all learn the mapping between GKT-engineered feature vectors and an outcome measure.

Re-casting the PPE in a GKTM To make the above more concrete, we now present an alternative implementation of the Predictive Performance Equation (PPE) (Jastrzembski, Gluck, & Gunzelmann, 2006; Walsh, Gluck, Gunzelmann, Jastrzembski, Krusmark, Myung, et al., 2018). However, first some background on the PPE and its structure: Given data from multiple users studying multiple stimuli, each user-stimulus pairing has their own time series, t. The values in t are subjected to two transformations that form the core of the PPE. First, we compute the *model time*, T, which is the weighted cumulative sum of elapsed time,

$$T_i = \sum_{i=1}^{N} w_i \cdot t_i \text{ with } w_i = \frac{t_i^{-x}}{\sum_{j=1}^{N} t_j^{-x}} \tag{1}$$

in which the weight, x, is customarily set to 0.6. Additionally, a *stability* term is computed as the lagged, cumulative mean of the inverse log lag times, Δ:

$$\text{stability} = \frac{1}{N-1} \cdot \sum_{i=1}^{N-1} \frac{1}{\ln(\Delta_i + e)} \tag{2}$$

In PPE, the model time and stability are combined into the model's forgetting term, which has two free parameters, b and m:

$$M = N^c \cdot T^{-d} \text{ with } d = b + m \cdot \text{stability} \tag{3}$$

The forgetting term is in turn combined with the learning term to yield the activation, M. The learning term, N^c, is a simple counter of the number of prior instances, N, to the power of a learning rate, c, which is usually fixed at 0.1. Finally, following ACT-R's assumptions, the activation is mapped onto performance using a logistic function with two additional parameters, τ and s.

In what we refer to as *vanilla* PPE below, we take a time series t as input, compute the various components of the model, and then estimate the four free parameters—b, m, τ, and s—of the nested structure using gradient descent.

Research on PPE suggests it outperforms competing cognitive models on a series of theoretical and applied criteria (we refer the interested reader to (Walsh, Gluck, Gunzelmann, Jastrzembski, Krusmark, Myung, et al., 2018) and (Walsh, Gluck, Gunzelmann, Jastrzembski, & Krusmark, 2018) for detailed background information). One limitation of the PPE is that it can only take a single input: timestamps. By re-casting the PPE—in what we think of as a "machine learning-ready" format—, we aim to preserve the cognitive model's insights by making them integral to the feature engineering step. At the same time, we want to divorce these features from the predictive engine and create the ability to ingest an arbitrary set of input features.

Related work

This section briefly reviews approaches that are relevant to the current work. Note that (Vie & Kashima, 2019) and (Pavlik et al., 2021) highlight how their—and thus by extension our—approaches build on earlier models, specifically (Bayesian) knowledge tracing, item response theory, and (variations of) additive and performance factor analysis.

Half-life regression (HLR) In the words of Settles and Meeder (2016), "HLR combines psycholinguistic theory with modern machine learning techniques". Their goal was to hard-code a theoretical assumption about human memory—an exponentially decaying forgetting curve—into a statistical model. This model predicts the probability of correctly recalling an item from memory by estimating the corresponding memory trace's *half-life* along with additional weights for "an arbitrarily large set of interesting features". The weights for each feature are estimated from empirical data using regularized regression (see section 3.3 in Settles and Meeder (2016) for details).

Generalized knowledge tracing (GKT) Pavlik et al. (2021) present a framework for learner modeling that they call GKT. The authors develop a symbolic notation system that allows the user to flexibly specify the components in a knowledge tracing model[1]. As the authors illustrate, many classic EDM models are special cases of their generalized framework. To demonstrate, the authors fit a range of models to 12 datasets and report that "no single learner model

[1] And the corresponding R package allows fitting such models: https://github.com/Optimal-Learning-Lab/LKT

was best in all cases, further justifying a broad approach that considers multiple learner model features and the learning context." Additional work from the same group accentuates the broad applicability to boot (Pavlik Jr & Eglington, 2021; Eglington & Pavlik Jr, 2019).

Knowledge tracing machines (KTM) Vie and Kashima (2019) point out that classic EDM approaches—such as additive factor and multidimensional item response theory models—are special cases of factorization machines. Specifically, they show that when the modeling process is conceived of as a general regression (or classification) problem, the difference between classic EDM models is primarily a matter of feature engineering. For example, the additive factor model (Cen, Koedinger, & Junker, 2006) includes a feature for the number of attempts; the performance factor analysis model (Pavlik Jr, Cen, & Koedinger, 2009) includes two separate features for correct and incorrect attempts. The authors point out how the underlying structure of these models can be extended by considering any number of additional features—which they call side information. They demonstrate this functionality by including embeddings learned from the data (Vie & Kashima, 2019, see Fig. 2) but any features—such as those in Duolingo (Vie, 2018)—can be handled. Recent work on the KTM-based DAS3H further illustrates the broad applicability of the approach (Choffin, Popineau, Bourda, & Vie, 2019, 2021; Choffin, Popineau, Bourda, et al., 2021).

Generalizing further The GKTM framework outlined above can thus be considered as an extension and slight reframing of both GKT and KTM. Specifically, we point out that their work focuses exclusively on the feature engineering side of the modeling pipeline. We believe that our suggestion of considering the predictive engine as an integral part of the model has the potential to further extend the predictive capacities of both GKT and KTM.

There is also great potential of cross-pollination between GKT and KTM: Both approaches are build on the same foundation of earlier work in the field and differ slightly in their feature engineering focus. The difference between their choice of predictive engines—probit and logistic regression—, however, is minimal. By pointing out the similarities between GKT and KTM and the potential of connecting their feature engineering efforts with the plethora of high-powered predictive engines available, we hope to generalize the modeling pipeline even further.

Additionally, the current work should also be considered a generalization of the PPE, in which we preserve its key components but cast it as a general purpose knowledge tracing model. The generalized PPE formulation can readily be combined with parts of both GKT (by using different learning terms or simpler decay terms, for example, Pavlik et al. (2021, see Table 1)) and KTM (by adding skill and stimuli biases and embeddings, for example). Importantly, this overcomes a major shortcoming of the current—"vanilla"—PPE in that it would also readily handle the features ingested by HLR, thereby greatly increasing its applied potential (Walsh, Gluck, Gunzelmann, Jastrzembski, & Krusmark, 2018).

Model comparison

The goal of the model comparison is twofold: contrast *vanilla PPE* with the re-cast, machine learning-ready version that has access to the same input features, and demonstrate the flexibility of the proposed framework by varying both the feature engineering steps and predictive engines.

Data

We will perform the model comparison on a dataset from an experimental study conducted in a sleep laboratory. The experiment is described in detail in (Walsh et al., 2022). In short, 72 participants studied digit-droodle paired associates across 11 sessions spread over three days. The task was to learn the correct two-digit response to a simple, abstract line drawing called a "droodle" (Nishimoto, Ueda, Miyawaki, Une, & Takahashi, 2010). The experimental within-subject manipulation assigned three digit-droodle pairs to each of 17 spacing conditions that distributed the same number of repetitions across sessions. They were exposed to each of the 51 stimuli during the intake meeting on the day prior to the study.

A visual overview of the study design is shown in Figure 1. It shows how the 11 sessions were distributed across three days; the number on each tile indicates the number of repetitions per stimulus per session, separately for each schedule. The third day only had a single session with two repetitions and will serve as the test data in the current application (7,242 instances; 9.4%). The data from the first two days (69,768 instances; 90.6%) will be used to train the models.

The data contain timestamped responses recorded from all users, including accuracy and response time, as well as additional labels such as the day (of day) of the study, the session number, and the schedule number.

Models

This section details the models we compare. In accordance with the framework outlined above, we will explain the feature engineering process and the predictive engine used for each model.

Vanilla PPE This model forms the baseline for our comparison and represents the "traditional" approach to fitting this type of cognitive model. As explained above, the time series for each user-stimulus pair is used to engineer the model's core features (the learning term, model time, and stability) before estimating its four free parameters (b, m, τ, and s) using gradient descent. The objective function minimizes the negative log loss between the model's fit and the empirical accuracy[2]. One set of parameters is estimated independently for each user (across all schedules and stimuli; Walsh et al. (2022)).

[2]This loss function is sometimes called (binary) cross entropy and is functionally identical to the binomial MLE (Myung, 2003).

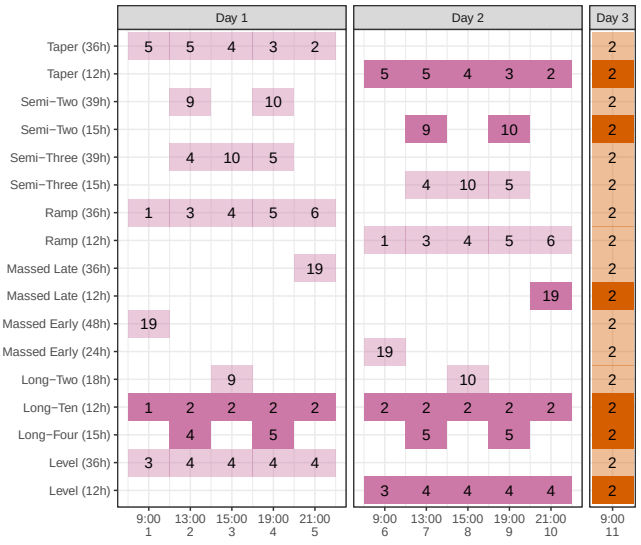

Figure 1: Repetition schedules in the data. For the model comparison, the purple sessions were used as training data and the orange session as test data. Subsequent plots will focus on the subset of schedules in darker colors.

GKTM$_{LR}$ This model is the direct re-cast of the vanilla PPE as a general purpose, "machine learning-ready" knowledge tracing model inspired by Pavlik et al. (2021) and Vie and Kashima (2019). The feature engineering included two aspects: computing the PPE model components (`lag` time, `model_time`, `stability`, and the learning term [`LT`]), and one-hot encoding the user identifiers as detailed by (Vie & Kashima, 2019, see their Fig. 1) (cf. with the user-level intercepts in Pavlik et al. (2021)). This results in a total of 77 features: One OHE'ed feature for each of the 72 users plus the raw `time` series plus the four PPE-based features.

As the predictive engine, we used a logistic regression (hence "LR"), which adds one additional parameter: the global bias, or intercept. All features except the learning term were standardized prior to fitting. A logistic regression was chosen to keep the comparison to vanilla PPE as direct as possible. This makes it similar to GKT and KTM as well. In fact, this model resembles a GKT model with intercepts for each learner and the features `linesuc` (or `logsuc`) and `ppe` (see Pavlik et al. (2021, Table 1)).

There are key differences between this model and vanilla PPE. First, the number of parameters differs substantially: vanilla PPE estimates $4 \times 72 = 288$ unique parameters; this model estimates 78. Second, the vanilla PPE estimates separate b and m parameters for each learner, which means that the weight assigned to the stability term in Eq. 3 can differ between learners. This model, on the other hand, only estimates a single coefficient for the `stability` feature across all users. Third, this model estimates specific coefficients (albeit across all users) for the other three PPE-derived features, which vanilla PPE does not. And fourth, vanilla PPE estimates two free parameters that control the logistic mapping from activation, M, to performance. This means that two users with the same activation value can have different performance profiles due to the flexibility afforded by user-specific τ and s parameters. In contrast, this model estimates a single intercept across all users and the coefficients estimated for each one-hot encoded user feature can be seen as a user-specific offset of the global bias.

GKTM$_{DT}$ This model uses the same input features as GKTM$_{LR}$ with one addition: the lagged response time (`lagged_rt`; in seconds). As a predictive engine, this model uses a single decision tree (hence "DT") as implemented in Therneau and Atkinson (2019). The response times were added under the assumption that they convey relevant information pertaining to memory strength (Madigan, Neuse, & Roeber, 2000; Sense & van Rijn, 2022) and were lagged to avoid data leakage. As such, this model serves as a minimal example of how additional features not anticipated by the PPE can easily be added alongside the PPE-based features. It also highlights the possible utility of highly interpretable predictive engines (see Figure 4).

GKTM$_{RF}$ This model contains all features of GKTM$_{DT}$ as well as all information apparent in Figure 1: the (time of) day, the session number, the schedule number, and the schedule type (e.g., "level", "long-four", etc.). This model uses a random forest as the predictive engine (hence "RF").

We choose a random forest to demonstrate how a more powerful ensemble of GKTM$_{DT}$-like models can make better use of the available features and achieve superior predictive performance[3]. What sets this model apart from the other models is that it contains a wider range of—both continuous and categorical—features and that the random forest can learn arbitrary (and potentially non-linear) relationships between features.

Approach

For the model comparison, we focus on the overall quality of the fit and the predictions made by the different models. We follow Vie and Kashima (2019) and others and present three metrics for fits and predictions: the classification accuracy (ACC), the area under the ROC curve (AUC), and the negative log likelihood[4] (NLL). Together, these metrics give a holistic impression of the models' performance.

Furthermore, we will zoom in on the differences between the schedules since those were experimentally manipulated

[3] Also, Bentéjac, Csörgő, and Martínez-Muñoz (2020) suggest that random forests require little to no hyper-parameter tuning to work well, which makes it attractive as an out-of-the-box predictive engine for the current demonstration (we used the implementation by Wright and Ziegler (2017) with default settings).

[4] The NLL is computed as $-\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(x_i) + (1-y_i) \cdot log(1-x_i)$, where y_i are the actual values and x_i are the predictions for that instance. To avoid $log(0)$, x_i values of exactly 0 or 1 were offset by 0.001. This means the maximal penalty is $-log(0.001) = 6.9$ and, consequently, the highest possible NLL value is the same—i.e., a model that predicts 0 for all correct responses and 1 for all incorrect responses.

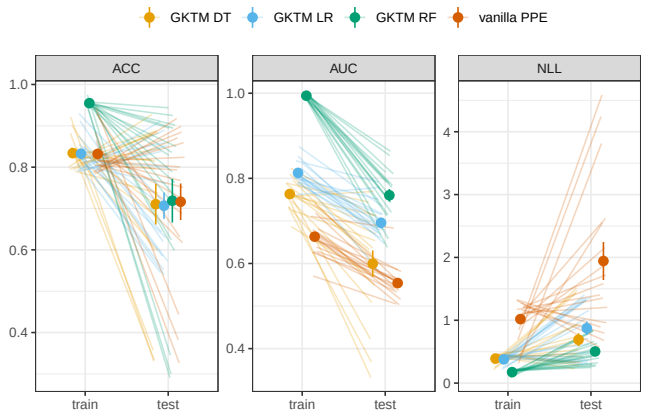

Figure 2: Model fits (train) and quality of predictions (test) for the three error metrics. Models are color-coded; lines correspond to schedules; each model's mean and standard error across schedules are indicated by circles and error bars, respectively.

in the study and induce the temporal variability that is our prime interest here. Due to space constraints, we will focus on a subset of six schedules. We chose the six schedules with the lowest mean absolute deviation between the predictions made by vanilla PPE and GKTM$_{LR}$, since that is the primary contrast we want to make here. The online supplement at https://osf.io/kugjn/ contains versions of all figures in which all schedules are shown.

Results of the model comparison

To reiterate, we have two goals: contrast vanilla PPE and GKTM$_{LR}$, and highlight the flexibility of the framework.

As a first step, we compare the models' ability to fit the training (purple sessions in Figure 1) and predict the test data (orange session). The results are shown in Figure 2. The lines represent the 17 schedules and colors correspond to the models. For the AUC and NLL, GKTM$_{LR}$ outperforms vanilla PPE—the ACC hardly differentiates between the four models. This suggests the differences between the two models played out to the advantage of the re-cast implementation in these data.

To contrast the two models further, Figure 3 shows the six schedules for which they produce the most similar predictions. Interestingly, GKTM$_{LR}$ predicts more extreme forgetting, which is apparent (during fitting) between Day 1 and 2 in Long-Four and -Ten and (for predictions) in the larger predicted decrease for repetition 20. This can lead to both over- (Long-Ten) and underestimation (Massed Late) of the actual amount of forgetting. However, both Figure 2 and https://osf.io/rgksp suggest GKTM$_{LR}$ makes better predictions overall.

Considering all four models in our assessment, we can see that all GKTM versions achieve better AUC and NLL values, with ACC values that vary drastically between schedules but

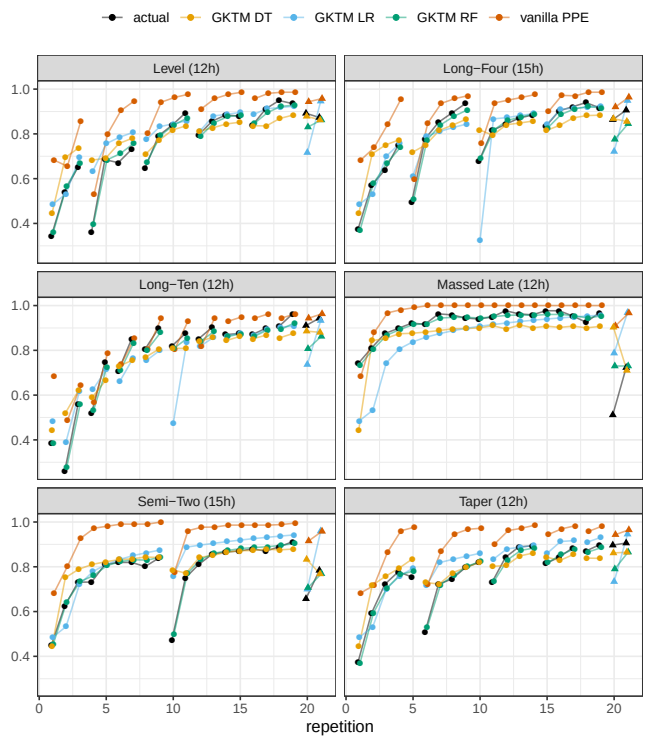

Figure 3: Fit (circles) and predictions (triangles) for all models; aggregated per repetition for six schedules. "Actual" are the empirical means; lines group repetitions within a session. See https://osf.io/bxjp9 for all schedules.

average out to be similar across all models. Somewhat surprisingly, even the GKTM$_{DT}$ model can compete with vanilla PPE, which suggests the lagged response times are very informative. The decision tree learned by the model is shown in Figure 4 and confirms this notion. Similarly, we can inspect the most powerful model: Figure 5 shows the feature importance for the random forest learned by GKTM$_{RF}$. Again, the `lagged_rt` and `lag` are most important. However, it draws on the model time and stability as well, even though the former is highly correlated with lag-time ($r = 0.866$; see https://osf.io/pm9zv for correlations between all numeric features)[5].

This exploration illustrates two things: (a) Even with a fixed set of engineered features, different predictive engines can achieve different results, and (b) there is a trade-off between more powerful predictive engines and interpretability. Interpretable machine learning tools are being developed (Molnar, 2019, 2020; Greenwell, Boehmke, & McCarthy, 2018) to hopefully make these models more relevant to researchers working in educational settings, where interpretability is often paramount.

[5]For GKTM$_{LR}$, we can also get an idea of the feature importance: coefficients for the PPE terms are $lag = -0.468$, $stability = 0.508$, $model\ time = 0.166$, and -0.030 for the raw elapsed $time$. Since these features were all standardized, their absolute values signal relative importance.

Discussion

To summarize, the preceding model comparison showed that the Predictive Performance Equation (PPE; (Walsh, Gluck, Gunzelmann, Jastrzembski, Krusmark, Myung, et al., 2018)) can be re-cast in what we termed a generalized knowledge tracing machine (GKTM). Leaning heavily on GKT (Pavlik et al., 2021) and KTM (Vie & Kashima, 2019), GKTM is a framework for flexible feature engineering that can be connected to a suite of predictive engines. We hope that the above has demonstrated the extent to which a model using PPE-derived features can easily be extended with additional features and the choice of a suitable predictive engine depends on the researcher's goal.

For work related to the PPE, we believe this new approach to using the model's insight brings a number of advantages. First, the outcome measure can be on any scale that suits the chosen predictive engine. This opens the door to modeling novel scenarios (e.g., one requiring multi-label classification). Second, the new implementation can handle any number of additional predictive features not foreseen by the original PPE (or similar cognitive models). Here, we showcased an example in which the response time was highly informative (see Figures 4 and 5). In many cases, one might realize that non-temporal features are generally more important (see, e.g., Table 4 in Settles, Brust, Gustafson, Hagiwara, and Madnani (2018)). In such cases, the third advantage is that this approach allows seamless integration of insights from a cognitive model into a general purpose machine learning model (Sense et al., 2021). And fourth, simpler GKTM$_{LR}$ and GKTM$_{DT}$ variants exemplified here are faster to fit and have fewer parameters than vanilla PPE. Furthermore, those parameters are arguably more interpretable. This is not necessarily true for more complex predictive engines but that might not be a problem if the goal is prediction rather than explanation (Shmueli et al., 2010; Yarkoni & Westfall, 2017). One reason the GKTM models outperform vanilla PPE is because they are fit to the entire training data at once; vanilla PPE considers each user's data independently. Versions of the PPE that use hierarchical Bayesian methods for parameter estimation (Collins, Gluck, Walsh, Krusmark, & Gunzelmann, 2016; Collins, Sense, Krusmark, & Jastrzembski, 2020; Walsh et al., 2022) have similar benefits but are computationally expensive.

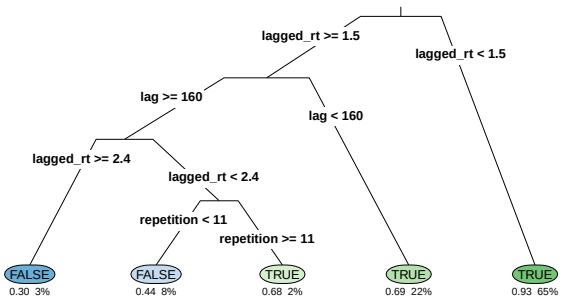

Figure 4: The decision tree learned by GKTM$_{DT}$.

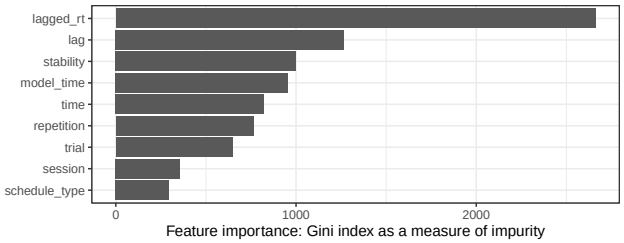

Figure 5: Feature importance for GKTM$_{RF}$.

There are a number of ways we would like to extend the current work. One is highlighted by the fact that the decision tree does fairly well in terms of prediction, even though it produces a theoretically implausible pattern (performance on repetition 21 is predicted to be lower than on 20). This underlines the importance of extending cognitive models' ability to use additional input features: the decision tree relies heavily on the (lagged) response time, which the PPE cannot. The fact that it uses lagged RT, then lag, then lagged RT again (Figure 4), also suggests that there is likely an interaction between the two top features in the random forest (Figure 5). Interpretable machine learning techniques (Molnar, 2019; Masis, 2021) should be explored to get a better grasp of feature interactions in complex predictive engines. Promising methods exist (Greenwell et al., 2018) that can be applied to models like the random forest used here or the other engines like recurrent neural networks (Lai, Wang, & Ling, 2021).

More generally, we would like to pitch the new formulation of the PPE against datasets used by Pavlik et al. (2021) and Vie and Kashima (2019) to get a better idea of the relative importance of PPE-based features as well as the differences in predictive accuracy for different predictive engines. For such efforts, adding OHE vectors for stimuli and knowledge components (like Vie and Kashima (2019) suggest), additional features based on embeddings (also Vie and Kashima (2019)), and whatever seems sensible off the menu offered by Pavlik et al. (2021) would be logical first steps for future feature engineering.

Given the wide range of possible input features, attention should also be paid to principled feature selection mechanisms. One obvious approach would be using regularization, as done by Settles and Meeder (2016).This is particularly attractive if one wants to use logistic regression as the predictive engine. Independent of the predictive engine, well-established filter, wrapper, and hybrid approaches are worth considering (Hua, Tembe, & Dougherty, 2009; Chandrashekar & Sahin, 2014).

Acknowledgments

This work was supported by a 711th Human Performance Wing Chief Scientist Seedling Grant and a research grant awarded by the Air Force Office of Scientific Research Sci-

ence of Information, Computation, Learning, and Fusion Program.

Data were wrangled in `R` (R Core Team, 2020) using `tidyverse` (Wickham et al., 2019); figures were created with `ggplot` (Wickham, 2016), `rpartplot` (Milborrow, 2020), and `vip`, (Greenwell & Boehmke, 2020). The modeling pipeline was implemented using `tidymodels` (Kuhn & Wickham, 2020).

References

Bentéjac, C., Csörgő, A., & Martínez-Muñoz, G. (2020). A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, 1–31.

Cen, H., Koedinger, K., & Junker, B. (2006). Learning factors analysis–a general method for cognitive model evaluation and improvement. In *International conference on intelligent tutoring systems* (pp. 164–175).

Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, *40*(1), 16–28.

Choffin, B., Popineau, F., Bourda, Y., et al. (2021). Extending adaptive spacing heuristics to multi-skill items. *Journal of Educational Data Mining*, *13*(3), 69–102.

Choffin, B., Popineau, F., Bourda, Y., & Vie, J.-J. (2019). DAS3H: modeling student learning and forgetting for optimally scheduling distributed practice of skills. *arXiv preprint arXiv:1905.06873*.

Choffin, B., Popineau, F., Bourda, Y., & Vie, J.-J. (2021). Evaluating DAS3H on the EdNet dataset. In *Aaai 2021-the 35th conference on artificial intelligence/imagining post-covid education with ai.*

Collins, M. G., Gluck, K. A., Walsh, M. M., Krusmark, M., & Gunzelmann, G. (2016). Using prior data to inform model parameters in the predictive performance equation. In *Cogsci.*

Collins, M. G., Sense, F., Krusmark, M., & Jastrzembski, T. S. (2020). Improving predictive accuracy of models of learning and retention through bayesian hierarchical modeling: An exploration with the predictive performance equation. In *Cogsci.*

Corbett, A. T., Koedinger, K., & Hadley, W. S. (2001). Cognitive tutors: From the research classroom to all classrooms. In *Technology enhanced learning* (pp. 215–240). Routledge.

Eglington, L. G., & Pavlik Jr, P. I. (2019). Predictiveness of prior failures is improved by incorporating trial duration. *Grantee Submission*, *11*(2), 1–19.

Greenwell, B. M., & Boehmke, B. C. (2020). Variable importance plots—an introduction to the vip package. *The R Journal*, *12*(1), 343–366. Retrieved from `https://doi.org/10.32614/RJ-2020-013`

Greenwell, B. M., Boehmke, B. C., & McCarthy, A. J. (2018). A simple and effective model-based variable importance measure. *arXiv preprint arXiv:1805.04755*.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

Hua, J., Tembe, W. D., & Dougherty, E. R. (2009). Performance of feature-selection methods in the classification of high-dimension data. *Pattern Recognition*, *42*(3), 409–424.

Jastrzembski, T. S., Gluck, K. A., & Gunzelmann, G. (2006). Knowledge tracing and prediction of future trainee performance. In *Interservice/industry training, simulation, and education conference* (pp. 1498–1508).

Kuhn, M., & Wickham, H. (2020). Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles. [Computer software manual]. Retrieved from `https://www.tidymodels.org`

Lai, Z., Wang, L., & Ling, Q. (2021). Recurrent knowledge tracing machine based on the knowledge state of students. *Expert Systems*, *38*(8), e12782.

Madigan, S., Neuse, J., & Roeber, U. (2000). Retrieval latency and "at-risk" memories. *Memory & cognition*, *28*(4), 523–528.

Masis, S. (2021). *Interpretable machine learning with python: Learn to build interpretable high-performance models with hands-on real-world examples*. Packt Publishing Ltd.

Milborrow, S. (2020). rpart.plot: Plot 'rpart' models: An enhanced version of 'plot.rpart' [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=rpart.plot` (R package version 3.0.9)

Molnar, C. (2019). *Interpretable machine learning*. (`https://christophm.github.io/interpretable-ml-book/`)

Molnar, C. (2020). *Interpretable machine learning*. Lulu. com.

Myung, I. J. (2003). Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology*, *47*(1), 90–100.

Nishimoto, T., Ueda, T., Miyawaki, K., Une, Y., & Takahashi, M. (2010). A normative set of 98 pairs of nonsensical pictures (droodles). *Behavior research methods*, *42*(3), 685–691.

Pavlik, P. I., Eglington, L. G., & Harrell-Williams, L. M. (2021). Logistic knowledge tracing: A constrained framework for learner modeling. *IEEE Transactions on Learning Technologies*.

Pavlik Jr, P. I., Cen, H., & Koedinger, K. R. (2009). Performance factors analysis–a new alternative to knowledge tracing. In *Proceedings of the 14th international conference on artificial intelligence in education* (pp. 531–538).

Pavlik Jr, P. I., & Eglington, L. G. (2021). Modeling the ednet dataset with logistic regression. *arXiv preprint arXiv:2105.08150*.

Pelánek, R. (2017). Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modeling techniques. *User Modeling and User-Adapted Interaction*,

27(3), 313–350.

R Core Team. (2020). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from https://www.R-project.org/

Sense, F., Jastrzembski, T. S., Mozer, M. C., Krusmark, M., & van Rijn, H. (2019). Perspectives on computational models of learning and forgetting. In *International conference on cognitive modeling*.

Sense, F., & van Rijn, H. (2022). Optimizing fact-learning with a response-latency-based adaptive system. *PsyArXiv preprint 10.31234/osf.io/chpgv*.

Sense, F., Wood, R., Collins, M. G., Fiechter, J., Wood, A., Krusmark, M., ... Myers, C. W. (2021). Cognition-enhanced machine learning for better predictions with limited data. *Topics in Cognitive Science*.

Settles, B., Brust, C., Gustafson, E., Hagiwara, M., & Madnani, N. (2018). Second language acquisition modeling. In *Proceedings of the thirteenth workshop on innovative use of nlp for building educational applications* (pp. 56–65).

Settles, B., & Meeder, B. (2016). A trainable spaced repetition model for language learning. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers)* (pp. 1848–1858).

Shmueli, G., et al. (2010). To explain or to predict? *Statistical science*, *25*(3), 289–310.

Therneau, T., & Atkinson, B. (2019). rpart: Recursive partitioning and regression trees [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=rpart (R package version 4.1-15)

Vie, J.-J. (2018). Deep factorization machines for knowledge tracing. *arXiv preprint arXiv:1805.00356*.

Vie, J.-J., & Kashima, H. (2019). Knowledge tracing machines: Factorization machines for knowledge tracing. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 750–757).

Walsh, M. M., Gluck, K. A., Gunzelmann, G., Jastrzembski, T., & Krusmark, M. (2018). Evaluating the theoretic adequacy and applied potential of computational models of the spacing effect. *Cognitive science*, *42*, 644–691.

Walsh, M. M., Gluck, K. A., Gunzelmann, G., Jastrzembski, T., Krusmark, M., Myung, J. I., ... Zhou, R. (2018). Mechanisms underlying the spacing effect in learning: A comparison of three computational models. *Journal of Experimental Psychology: General*, *147*(9), 1325.

Walsh, M. M., Krusmark, M., Jastrzembski, T., Hansen, D. A., Honn, K. A., & Gunzelmann, G. (2022). Enhancing learning and retention through the distribution of practice repetitions across multiple sessions. *PsyArXiv Preprint: 10.31234/osf.io/dmf4p*.

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York. Retrieved from https://ggplot2.tidyverse.org

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., ... Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. doi: 10.21105/joss.01686

Wright, M. N., & Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, *77*(1), 1–17. doi: 10.18637/jss.v077.i01

Yarkoni, T., & Westfall, J. (2017). Choosing prediction over explanation in psychology: Lessons from machine learning. *Perspectives on Psychological Science*, *12*(6), 1100–1122.