Fret Ferret: Driving Educational Games with Contextual Multi-Armed Bandits to Accelerate Human Learning

DOUGLAS MASON, Koyote Science, LLC, USA
DAN COTE, Koyote Science, LLC, USA

This work introduces the world's most robust software trainer for the guitar, called *Fret Ferret*, which utilizes the contextual multi-armed bandit class of algorithms, as well as implicit signals from the user, to customize lessons to each user automatically and in real time, avoiding the need to tweak a large selection of game parameters in order to obtain a desired difficulty level. We discuss the consequences of using these algorithms to drive new styles of computer-human interaction that can accelerate our learning process and help us reach new mental achievements. We elaborate on the details of our algorithms, how they inform our user interface design, and how we address the difficulty of scaling the development of a large number of machine learning models to adapt to many mini games and each user individually. Finally, we address future work that can further augment the performance and capabilities of our software.

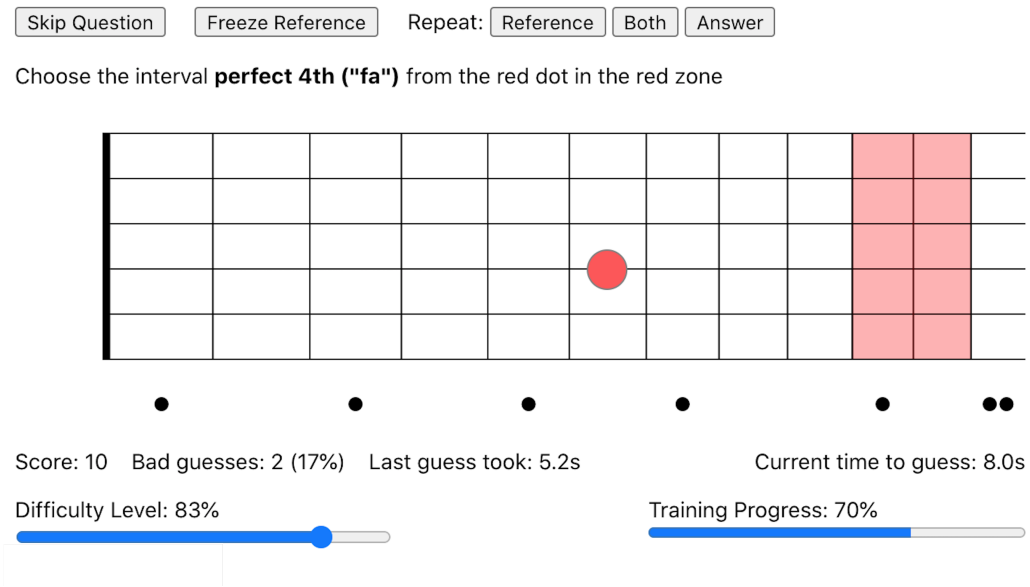

Fig. 1. Example user interface for identifying intervals visually on the guitar fretboard, with the questions determined by a contextual multi-armed bandit aiming for moderate-to-high difficulty and that has been partially trained.

1 INTRODUCTION

Software for training guitar players has generally been limited to random selections of questions with a focus on correctness over speed[1–3]. However, users quickly get bored when such systems ask too many questions they know

Authors' addresses: Douglas Mason, douglas@koyotescience.com, Koyote Science, LLC, , San Francisco, California, USA, 94114; Dan Cote, dan@koyotescience.com, Koyote Science, LLC, , San Francisco, California, USA, 94114.

by heart, or ask questions that are outside of their abilities. To address this limitation, the common solution has been to offer a large variety of randomization parameters for the user to adjust in order to eliminate questions as they become easier to answer.

Here we introduce our own guitar training software *Fret Ferret* to show how we have marshaled a contextual multi-armed bandit algorithm[6, 17] to quickly learn from implicit user feedback during early trials. This allows us to hide the randomization parameters and automatically alter question selection to optimize for a desired difficulty level. For example, if a user desires high difficulty, the system will select questions which are likely to take the longest to answer and focus on their weak spots. If this proves too challenging to stay enjoyable, the user can reduce the difficulty by adjusting one intuitive parameter rather than many confusing ones (see Figures 1 & 2).

The multi-armed bandit class of algorithms[6, 17] were first introduced in the 1980s, but only recently have technological developments and improved methods allowed us to implement them easily and cheaply. In the last decade, they have exploded onto the education research scene[11, 22, 31, 37, 45] as an ideal method for automatically tailoring questions to students without human intervention, which can greatly reduce labor costs and increase access. While bandits optimize for an immediate response, such as correctness or speed of the question at hand, reinforcement learning algorithms additionally consider how sequences of decisions can lead to longer-term responses over multiple questions, such as how long the student continues to use the system, which have also been explored in the literature[5, 34, 38]. Similarly, multi-armed bandits[23, 32] and reinforcement learning[40, 46] have been explored for game and interface design, among many other applications[8, 21].

Beyond the literature we have examined on multi-armed bandit and reinforcement learning applications in educational games, this article contributes the following:

- Our reward signal is based on **time-to-completion** instead of answer correctness.
- We **enhance exploration** using intuitive parameters that designers and users can adjust during gameplay.
- We address the challenge of **feature engineering** and hyperparameter tuning in an online fashion that is inconspicuous to the user and requires no manual oversight.
- We combine state-of-the-art algorithms with **multiple modes of interactions** for a comprehensive educational experience

All of these contributions have been born out of necessity for building a large corpus of auto-personalizing games quickly which can respond to the unique needs of advanced musical education.

Moreover, in this article we will discuss how merging multi-armed bandits with novel interaction mechanisms presents unique challenges for interface and machine learning model design, with each informing the other. For example, the bias-variance trade-off[7] in machine learning manifests itself by determining how long it takes for a game to correctly predict the difficulty of each question for the user, making model tuning critical to finding the right balance. This challenge arises because we are focused on building personalized curricula, rather than optimizing one curriculum over multiple students. We discuss our solution, which we call *HyperTune*, and which makes it easy to deploy the most enjoyable, effective educational tool possible with the latest advances in machine learning.

2 WHY USE MULTI-ARMED BANDITS?

Multi-armed bandits are commonly used in commercial technology to optimize websites[44] as a replacement for A/B testing of different presentation options, such as choosing between two headlines for a newspaper article[12]. These algorithms automatically navigate the exploration/exploitation trade-off by first selecting options randomly (exploration)

then honing in on the most promising ones (exploitation) in a manner that reduces the regret (the difference between real-world performance and ideal performance). Because of their data efficiency, bandits are ideal for personalization in situations where there are few opportunities to learn a user's preferences.

In our application, we use a multi-armed bandit to optimize for a desired difficulty level as evaluated implicitly by time-to-completion: high difficulty correlates to questions that take the longest to answer, and vice versa. Difficulty is measured as the percentile of random questions whose predicted time-to-completion are shorter than the measured time-to-completion for the given question. The bandit trains against the time-to-completion target, and questions are ranked according to the rank score. Given a prescribed difficulty level D, question Q with predicted time-to-completion $T(Q)$, and time-to-completion percentile $P(Q;\boldsymbol{Q})$ for Q among all questions $\boldsymbol{Q}$, we write the rank score as

$$R(Q) = \text{abs}\left[P(Q;\boldsymbol{Q}) - D\right] \tag{1}$$

Ranking is unaffected by additive shifts or multiplicative constants. Note that because of the additional restraint of matching a desired difficulty level, our approach does not match the bandit formalism strictly, except when D is set to 100%. Rather, we have decoupled the target that the bandits are trained against from the ranking procedure used to select questions, but the desired behavior remains the same.

We chose time-to-completion as a metric for many reasons, beginning with the fact that it is very difficult to cheat, making it one of the most reliable measures of cognition in the psychology literature, as highlighted by the Implicit Association Test[18]. Another benefit is that our target audience of musicians desire not only to be correct, but to recall information as quickly as possible. Moreover, optimizing computer-human interaction to consistently operate at the edge of the user's abilities is the hallmark of creating "flow" states that are intrinsically enjoyable to the user and critical for creative development[13], while keeping users engaged and excited to keep using our system. Without automated difficulty matching, users are forced to answer many questions that are beneath or above their level, which becomes boring or frustrating, leading to churn.

3 STRATEGY CONSIDERATIONS

Bandits are necessary in environments where data is expensive to collect – in our case, because each question requires time for the user to respond, and because users have limited patience. When a bandit makes predictions and acts on them with a given strategy, it not only samples the most promising options, it also learns the most about the domain of data where those options reside. Unlike traditional supervised machine learning models, which attempt to reflect the full domain of possible inputs in an unbiased fashion, bandits are intentionally blind to unexplored regions in the input space since it is costly to obtain information about them. In our application, this behavior manifests itself when the user changes the desired difficulty level, since the bandit must re-engage exploration, and will make some wrong guesses as it learns about the new data domain, until it is confident it can exploit at the new difficulty level again.

There are many strategies available to automate this process, chief among them are (1) the "epsilon-greedy" strategy[44], which only uses a prediction model for selecting options for a given fraction of questions, usually around 80%, and selects randomly otherwise, (2) the Upper Confidence Bound (UCB) strategy[20], which ranks options based on their predicted outcomes in addition to their uncertainties, and (3) Thompson sampling[10, 14, 36], in which the prediction model is replaced by a distribution of models, one of which is sampled to perform predictions for each question that is asked. Model predictions, and their confidence intervals, are transformed by Equation 1 before being acted upon any given strategy. For our application, we use Thompson sampling because it only introduces one

hyperparameter to tune, the number of prediction samples, which is generally set to 100, and it has shown stronger performance over epsilon-greedy and UCB in similar contexts[35].

For the prediction model, we use a custom implementation of Bayesian linear regression[9, 27] (BLR), which provides us model distributions and Bayesian sampling to enable Thompson sampling. BLR has the benefit of only requiring $O\left(N^2_{\text{features}}\right)$ storage and $O\left(N^3_{\text{features}}\right)$ computation, where N_{features} is the number of features in the model. Unlike other models like Gaussian processes or nearest neighbors regression, the computational and storage costs BLR does not scale with the amount of data it has been trained on, thus making it an incremental, online, or streaming algorithm with good performance guarantees. Moreover, because BLR is a convex optimization problem, it is always guaranteed to converge regardless of our hyperparameters, unlike stochastic gradient descent and neural networks that are popular in deep learning but which can get stuck in local extrema in their optimization landscape, requiring extensive hyperparameter tuning. BLR is also wildly more efficient for prediction than random forests, which is critical for latency-sensitive tasks like interactive games. Most importantly, these properties allow BLR to introduce only one new hyperparameter to tune: the regularization constant, which can generally be left to a standard value such as 1/100th of the vector norm of the inputs after a set number of inputs have been observed.

With *Fret Ferret*, users specify a difficulty level between 0% and 100%, with the default set to 80%, and which is interpreted as a difficulty percentile of available questions. We found that only choosing the question at that exact percentile, which is the traditional bandit setup, led to poor experiences with insufficient variety to keep the user engaged, even though Thompson sampling partially randomizes its selection process anyways. Thus, we artificially increased the variety of our question by selecting not only the most-closely-ranked question, but also others selected at random from a 10% population pool of questions with ranks close to the desired percentile. Anecdotal evaluations found this approach to be much more enjoyable and led to longer sessions.

4 USING *HYPERTUNE* TO SCALE MODEL DEVELOPMENT OVER MANY GAMES

The guitar trainer software we designed is a collection of many mini-games, each requiring a different prediction model to run its associated multi-armed bandit. The list of available games comes from the cross product of key skills we aim to train. On the one hand, players must be able to locate notes which are provided by name to gain a good *absolute* understanding of the fretboard, as well as musical intervals which can be written or played musically to gain a good *relative* understanding. On the other hand, we force users outside of their comfort zone by asking them to identify these locations under various restrictions which mimic common playing situations: (1) anywhere on the fretboard, (2) on a specified string, or (3) within a specified region of contiguous frets, usually 3 but other sizes are possible. Higher-level games include spelling out all notes in a chord or scale, starting with either a root name or location indicated visually on the fretboard. At the highest level, a chord progression can be performed by our web app, with the corresponding notes shown on the app's fretboard, while the performer improvises their own melodies which are also visually indicated on the same fretboard, to identify when the performer is playing "within" the chord and when they are playing "outside". The chord progressions can be defined by the user and even include key changes, making this interaction mode ideal for developing jazz fluency.

We have provided a syllabus of 32 such games in increasing order of sophistication to help the user along their journey. While this breadth enables us to work with players possessing a wide range of education levels, the large number of mini games introduces the burden of engineering the features used to train the models that run the bandits powering each game. Notably, as more features are added – for example, not only encoding which fret the answer note is played on, but also the string – the model begins with greater uncertainty in its predictions, requiring more

trials before it can refine those uncertainties. The bias-variance trade-off expresses itself in our games such that models with more features are more capable of finding diamonds in the rough, such as a particularly challenging fretboard location or note name for the user. Conversely, models with fewer features quickly pick out gestalt features, such as which strings or fretboard areas the user has difficulty with, but are incapable of refining their predictions further. Early efforts tuned the feature sets through trial-and-error to find those that allow the model to learn as quickly as possible while still capturing the most important aspects for predicting difficulty.

However, feature engineering is arduous and prone to error, making such systems difficult to scale or for new developers to utilize. Moreover, the best feature engineering for a given game may be specific to each user. For this reason, we incorporate the *HyperTune* system into our model, by which the developer can specify M sets of model parameters, which include parameters for the feature engineering used, that the system may consider. For each parameter set, N copies of the model are trained on a different sampling of $\frac{\max(N,5)-1}{\max(N,5)}$ of the full dataset, with the remaining $\frac{1}{\max(N,5)}$ of data held out for validation and stored. We apply the max operator $\max(N, 5)$ to ensure that at most 20% of data is reserved for validation. N-fold cross-validation is performed at regular intervals for each parameter set, whereby the associated model's performance over the validation data is averaged over the N copies. The set with the best average performance is selected, and an associated model that has been trained on all the data is silently deployed to the user.

This setup allows us to perform *HyperTune* in a streaming manner, with training latency and storage increased only by the constant factor $N \times M$, where $N = 1$ is often sufficient. However, the user only experiences the inference latency period, which is unaffected by *HyperTune*, so that they are oblivious to the work happening behind the scenes serving them the best possible model available. In addition, by using a reservoir sampler[41] to store the validation data, we can impose an upper limit on storage costs while maintaining an accurate representation of the data that has been seen.

Because the data we use to train our models is biased towards questions favored by the bandit that collected the data, it is helpful to compensate for this bias using *inverse probability weighting*[25]. In this approach, the probability that the currently-engaged bandit selected a given question indexed by i is written as π_i, and we weight the data used to train all our models by $\text{IPW}_i = \frac{1}{\pi_i}$. This ensures that feedback for questions which had a low likelihood being selected have a larger impact on subsequent decisions. Because of the inverse proportionality, numerical instabilities can arise when actions are given very small probabilities, but this is ameliorated by adding artificial exploration through an epsilon-greedy policy or other strategies, such as selecting questions from a ranked pool. Further elaboration, called *augmented inverse probability weighing* or *doubly-robust estimation* can also be considered[16].

Without *HyperTune*, a single model can be measured in terms of "progress", a term indicating how many varied samples have been encountered that lead to confident predictions of question difficulty. How this value is determined is left up to the developer, and is subjective in nature. For *Fret Ferret*, we define 100% progress as having seen at least 5 examples featuring each possible feature value for categorical variables, such as 5 examples for each string and 5 examples for each fret. Additional measures can be made for continuous variables, for example, by using domain knowledge to bin inputs to create equivalent categorical variables.

Employing *HyperTune* means that our models are continuously trained and swapped as they learn, so that like a car shifting to higher gears, the progress presented to the user drops down briefly as the more feature-rich is swapped in. However, we found that users respond better to a monotonically increasing measure of training progress as a goal to reach, and that simple heuristic measures like the one proposed here were more predictable and rewarding than direct measure of prediction uncertainty which can fluctuate substantially. For our user interface, we kept the progress bar for

the hand-tuned feature engineering we found worked well in general, but future measures of progress may be explored that are more consistent with the underlying models being used.

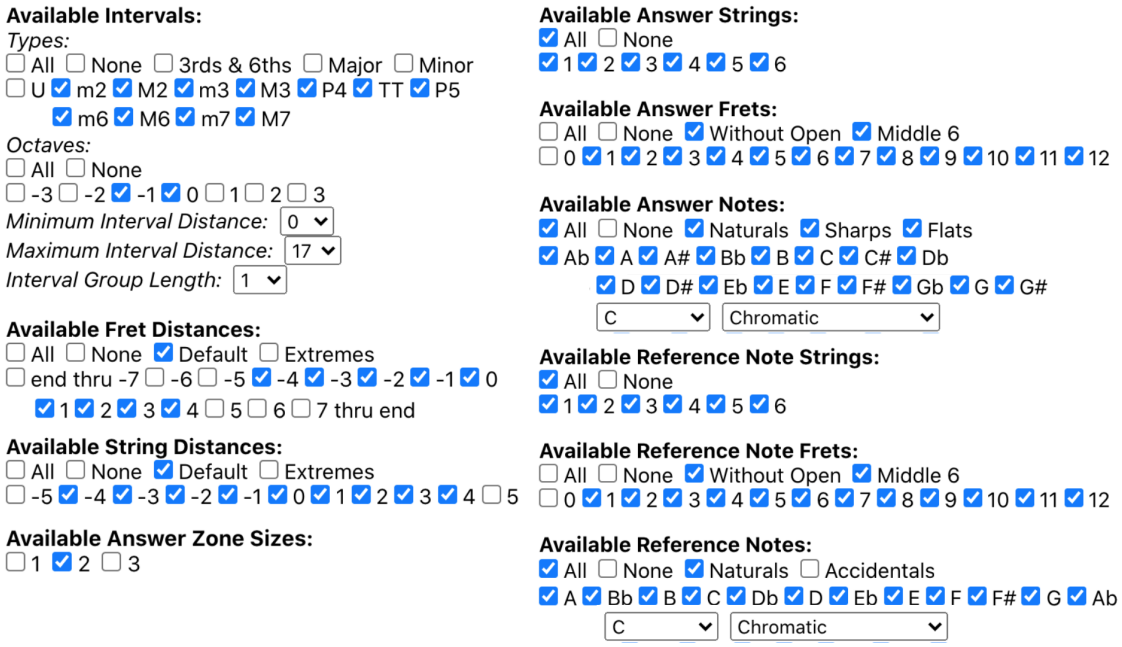

Fig. 2. The randomization parameters users otherwise need to adjust for the game in Figure 1. This image is taken from actual gameplay and was the original approach used before the algorithms in this article were implemented.

5 MODES OF INTERACTION

Working with the field of music provides a wide variety of ways to interact with software, in part thanks to the proliferation of portable touch-screen devices with high-quality speakers and microphones that can even work simultaneously without overstepping each other. At the moment, *Fret Ferret* provides the following avenues for interacting with the software:

- The user can click with a mouse or their finger on the fret position and string corresponding to their answer
- The user can play the corresponding note, either on their own guitar or any instrument of their choosing, even by humming the note
- The user can play the corresponding note on a MIDI instrument connected to the device

While working directly with the GUI is intuitive and always-available, the last two modes open up exciting possibilities with ear-training and transferring knowledge directly to the instrument, since the user can work with their native instrument instead of mapping one's knowledge to a GUI. These modes also make *Fret Ferret* a viable trainer for any instrument, but without the added restrictions of forcing the user to play notes on a given string or fret region which are specific to the guitar.

With modern pitch detection software[19], accurate predictions of pitch can be made within a few hundred milliseconds. This means that while fast solos and runs are difficult to accommodate, moderate speed and rhythm can be

maintained throughout the training session. Combining pitch detection with audio cues, an entirely-audio game can be played. Users of *Fret Ferret* that have worked with this mode have reported not just improved abilities at the guitar, but also improved listening skills that help the user identify the scale degrees (the "do-re-mi" label) of music that they listen to in real-time, helping them become better composers and improvisers.

A single note, such as E, can be played in many places on the guitar. Audio- and MIDI-based interaction modes are unable to pick up on the specific fret position or string that the user played. However, such interactions could be facilitated by an electronic instrument that delineates those distinctions[26], or by machine learning models that specialize in interpreting audio and/or video signals for this purpose. More experimental solutions include using models to predict the most likely location where the user played the note, which has been explored in automated tablature transcriptions[30] and predicting the location of mouse clicks using long-short-term memory models[42]. For our purposes, however, such elaborations have been unnecessary because there are few reachable notes from a given position, and the likelihood of choosing the same note outside of the restrictions is low (1/12 or 8%). Moreover, this error has little impact on the sound that is actually produced. While pitch detection for single notes is a mostly-solved problem with many viable alternatives[19] ready to be used, pitch detection for multiple notes at once is an ongoing field of research[15], and could be used in the future to add chord-playing without the requirement to hit each note individually.

6 CONCLUSIONS AND FUTURE WORK

The *Fret Ferret* guitar training software introduced here is unique for using implicit feedback and machine learning to tune its questions to the user automatically, radically simplifying the interface design so that only a single parameter needs to be controlled. This allows for easy adoption and a comfortable learning experience that eliminates wasted time as the student reaches towards their goals. Incorporating the *HyperTune* system, and being aware of the various impacts that our algorithm design have on the user experience, have allowed *Fret Ferret* to far exceed our expectations. By demonstrating the system and identifying key stumbling blocks, we hope to encourage others to create more educational tools driven by artificial intelligence that improve our human intelligence. There are, however, many ways we hope to improve our system and *Fret Ferret* further.

One limitation to the bandit framework is that it treats every question on its own without considering how sequences of questions can lead to harder or easier experiences for the user. For example, if a question is repeated, it will likely be a lot easier the second time around. One general solution to this problem is to add historical data to the feature set, such as providing the same features used to for the current question in order to describe previous questions in the queue. The most general solution may also add possible interactions between the same features for the different questions, such as whether the previous questions featured the same strings or frets. A major drawback to this class of solutions, however, is that models trained with such large feature sets fail to learn quickly and impose higher resource overhead in every aspect of computation. Alternatively, heuristics can be used to filter out question from the ranking pool, such as those which were recently asked, which is used in *Fret Ferret*. However, the most principled approach to handling sequential planning is to use the full force of reinforcement learning.

The most natural extension from bandits into the broader tool set of reinforcement learning is to graduate from bandits to Q-learning[39] by training against not just the immediate signal (time-to-answer for the given question), but an aggregate of future rewards signals, such as the average time it takes the user to answer the following N questions, or the maximum time to answer a question within that set. Similarly to inverse probability weighting, we also must adjust these rewards using importance sampling in the Thompson-sampling setting, as discussed in Section 4.3, "The

connection between multi-armed bandits and Q-learning" in [28]. Further development may work with the policies themselves in an approach known as *policy optimization*[43]. We hope to explore all these possibilities in the near future.

Fret Ferret is run by federations of models in the *HyperTune* system, whereby each federation is unique to each mini game and user, and therefore its models start from scratch whenever they encounter a new user or a user begins a new game. This arises because we are focused on building personalized curricula, rather than optimizing one curriculum over multiple students, although the algorithms discussed here can be easily adapted to either setting. However, this limitation poses the question: What if we could address both optimization problems simultaneously?

Merging global with personal optimization is a broad goal of the recommender systems and automated personalization literature[24, 47], which has also explored incorporating multi-armed bandit and reinforcement learning algorithms[4, 48, 50]. We have considered a few approaches for future development. One trains a global federation of models for all users and each mini game, then creates a copy of each global model to fine-tune it to each user, in a process called transfer learning[33, 49]. Another uses predictions from the global model as an additional input to an associated personalized model, allowing the personalized model to learn a gate parameter that determines how much to weigh the contribution from the global model[29]. Because the gate parameter is interpretable, it gives the developer or user control over the influence of global predictions even as the bandit learns the ideal value. This control can be introduced through the use of prior beliefs or by direct manipulation, with the drawback that the cumulative reward may not be maximized.

Lastly, the interactive demonstration presented here has not been subjected to user experience studies, by which we can measure user progress and retention as a function of the parameters we use to define our system, such as the impact of *HyperTune*, using different feature engineering approaches, etc., and we would like to pursue this angle as resources allow. In the meantime, *Fret Ferret* is free and open to the public, along with our 32-course syllabus, and can be accessed at http://fretferret.com.

REFERENCES

[1] 2020. *FaChords Guitar.* https://www.fachords.com/guitar-learning-software/

[2] 2020. *Guitar Notes Fretboard Trainer.* https://www.guitarorb.com/guitar-notes

[3] 2020. *Strong Apps.* http://www.strongapplications.com

[4] M Mehdi Afsar, Trafford Crump, and Behrouz Far. 2021. Reinforcement learning based recommender systems: A survey. *arXiv preprint arXiv:2101.06286* (2021).

[5] Jonathan Bassen, Bharathan Balaji, Michael Schaarschmidt, Candace Thille, Jay Painter, Dawn Zimmaro, Alex Games, Ethan Fast, and John C Mitchell. 2020. Reinforcement learning for the adaptive scheduling of educational activities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.

[6] Donald A. Berry and Bert Fristedt. 1985. *Bandit problems : sequential allocation of experiments / Donald A. Berry, Bert Fristedt.* Chapman and Hall London ; New York. viii, 275 p. : pages.

[7] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag, Berlin, Heidelberg.

[8] Djallel Bouneffouf, Irina Rish, and Charu Aggarwal. 2020. Survey on applications of multi-armed and contextual bandits. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[9] George EP Box and George C Tiao. 2011. *Bayesian inference in statistical analysis*. Vol. 40. John Wiley & Sons.

[10] Olivier Chapelle and Lihong Li. 2011. An Empirical Evaluation of Thompson Sampling. In *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger (Eds.), Vol. 24. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2011/file/e53a0a2978c28872a4505bdb51db06dc-Paper.pdf

[11] Benjamin Clement, Didier Roy, Pierre-Yves Oudeyer, and Manuel Lopes. 2013. Multi-armed bandits for intelligent tutoring systems. *arXiv preprint arXiv:1310.3174* (2013).

[12] Anna Coenen. 2020. *How The New York Times is Experimenting with Recommendation Algorithms.* https://open.nytimes.com/how-the-new-york-times-is-experimenting-with-recommendation-algorithms-562f78624d26

[13] Mihaly Csikszentmihalyi. 1991. *Flow: The Psychology of Optimal Experience.* Harper Perennial, New York, NY. http://www.amazon.com/gp/product/0060920432/ref=si3_rdr_bb_product/104-4616565-4570345

[14] Dean Eckles and Maurits Kaptein. 2014. Thompson sampling with the online bootstrap. *arXiv preprint arXiv:1410.4009* (2014).

[15] Anders Elowsson. 2018. Polyphonic Pitch Tracking with Deep Layered Learning. *CoRR* abs/1804.02918 (2018). arXiv:1804.02918 http://arxiv.org/abs/1804.02918

[16] Michele Jonsson Funk, Daniel Westreich, Chris Wiesen, Til Stürmer, M. Alan Brookhart, and Marie Davidian. 2011. Doubly Robust Estimation of Causal Effects. *American Journal of Epidemiology* 173, 7 (03 2011), 761–767. https://doi.org/10.1093/aje/kwq439 arXiv:https://academic.oup.com/aje/article-pdf/173/7/761/17338964/kwq439.pdf

[17] J. C. Gittins. 1989. *Multi-armed Bandit Allocation Indices.* Wiley, Chichester, NY.

[18] Anthony G Greenwald, Debbie E McGhee, and Jordan L. K Schwartz. 1998. Measuring Individual Differences in Implicit Cognition: The Implicit Association Test. *Journal of personality and social psychology* 74, 6 (1998), 1464–1480.

[19] Peter Hayes. 2020. *pitchfinder.* https://github.com/peterkhayes/pitchfinder

[20] T.L Lai and Herbert Robbins. 1985. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics* 6, 1 (1985), 4–22. https://doi.org/10.1016/0196-8858(85)90002-8

[21] Yuxi Li. 2017. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274* (2017).

[22] Yun-En Liu, Travis Mandel, Emma Brunskill, and Zoran Popovic. 2014. Trading Off Scientific Knowledge and User Learning with Multi-Armed Bandits.. In *EDM*. 161–168.

[23] J Derek Lomas, Jodi Forlizzi, Nikhil Poonwala, Nirmal Patel, Sharan Shodhan, Kishan Patel, Ken Koedinger, and Emma Brunskill. 2016. Interface design optimization as a multi-armed bandit problem. In *Proceedings of the 2016 CHI conference on human factors in computing systems*. 4142–4153.

[24] Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. 2015. Recommender system application developments: a survey. *Decision Support Systems* 74 (2015), 12–32.

[25] Mohammad Ali Mansournia and Douglas G Altman. 2016. Inverse probability weighting. *BMJ* 352 (2016). https://doi.org/10.1136/bmj.i189 arXiv:https://www.bmj.com/content/352/bmj.i189.full.pdf

[26] Karola Marky, Andreas Weiß, Florian Müller, Martin Schmitz, Max Mühlhäuser, and Thomas Kosch. 2021. *Let's Frets! Mastering Guitar Playing with Capacitive Sensing and Visual Guidance.* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411763.3451536

[27] Douglas Mason. 2021. *Gaussian processes and equivalent Bayesian linear regressions.* https://doi.org/10.5281/zenodo.5828879

[28] Douglas Mason. 2021. *Real-World Reinforcement Learning.* https://doi.org/10.5281/zenodo.5828870

[29] Douglas Mason. 2022. *The Koyote Science, LLC, Approach to Personalization and Recommender Systems.* https://doi.org/10.5281/zenodo.455689261

[30] Elias Mistler. 2017. Generating guitar tablatures with neural networks. *Master of Science Dissertation, The University of Edinburgh* (2017).

[31] John Mui, Fuhua Lin, and M Dewan. 2021. Multi-armed Bandit Algorithms for Adaptive Learning: A Survey. In *International Conference on Artificial Intelligence in Education*. Springer, 273–278.

[32] Santiago Ontanón. 2013. The combinatorial multi-armed bandit problem and its application to real-time strategy games. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference.*

[33] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2009), 1345–1359.

[34] Siddharth Reddy, Sergey Levine, and Anca Dragan. 2017. Accelerating human learning with deep reinforcement learning. In *NIPS'17 Workshop: Teaching Machines, Robots, and Humans*. 5–9.

[35] Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. 2020. A Tutorial on Thompson Sampling. arXiv:1707.02038 [cs.LG]

[36] Steven L. Scott. 2010. A modern Bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry* 26, 6 (2010), 639–658. https://doi.org/10.1002/asmb.874 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/asmb.874

[37] Avi Segal, Yossi Ben David, Joseph Jay Williams, Kobi Gal, and Yaar Shalom. 2018. Combining difficulty ranking with multi-armed bandits to sequence educational content. In *International conference on artificial intelligence in education*. Springer, 317–321.

[38] Adish Singla, Anna N Rafferty, Goran Radanovic, and Neil T Heffernan. 2021. Reinforcement Learning for Education: Opportunities and Challenges. *arXiv preprint arXiv:2107.08828* (2021).

[39] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction.* MIT press.

[40] Julian Togelius and Jurgen Schmidhuber. 2008. An experiment in automatic game design. In *2008 IEEE Symposium On Computational Intelligence and Games*. Citeseer, 111–118.

[41] Jeffrey S. Vitter. 1985. Random Sampling with a Reservoir. *ACM Trans. Math. Softw.* 11, 1 (mar 1985), 37–57. https://doi.org/10.1145/3147.3165

[42] Datong Wei, Chaofan Yang, Xiaolong (Luke) Zhang, and Xiaoru Yuan. 2021. *Predicting Mouse Click Position Using Long Short-Term Memory Model Trained by Joint Loss Function.* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411763.3451651

[43] Lilian Weng. 2018. Policy Gradient Algorithms. *lilianweng.github.io/lil-log* (2018). https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html

[44] J.M. White. 2012. *Bandit Algorithms for Website Optimization: Developing, Deploying, and Debugging.* O'Reilly Media. https://books.google.com/books?id=xnAZLjqGybwC

[45] Joseph Jay Williams, Juho Kim, Anna Rafferty, Samuel Maldonado, Krzysztof Z Gajos, Walter S Lasecki, and Neil Heffernan. 2016. Axis: Generating explanations at scale with learnersourcing and machine learning. In *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*. 379–388.

[46] Haifeng Zhang, Jun Wang, Zhiming Zhou, Weinan Zhang, Ying Wen, Yong Yu, and Wenxin Li. 2017. Learning to design games: Strategic environments in reinforcement learning. *arXiv preprint arXiv:1707.01310* (2017).

[47] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.

[48] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 95–103.

[49] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2020. A comprehensive survey on transfer learning. *Proc. IEEE* 109, 1 (2020), 43–76.

[50] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 2019. Reinforcement learning to optimize long-term user engagement in recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2810–2818.