Deep Bayesian Bandits Showdown

Carlos Riquelme, George Tucker, Jasper Snoek
Google Brain
{rikel, gjt, jsnoek}@google.com

Abstract

Recently, deep reinforcement learning has made significant strides in performance on applications such as Go and Atari games. However, practical methods for balancing exploration and exploitation in complex domains is still largely unsolved. Thompson Sampling and its extension to reinforcement learning provide an elegant approach that only requires access to posterior samples. At the same time, advances in approximate Bayesian methods have made posterior approximation for neural networks practical. As a first step towards understanding the impact of approximate posteriors on Thompson Sampling, we compare approximate posterior sampling methods combined with Thompson Sampling in a series of contextual bandit problems.

1 Introduction

Balancing exploration and exploitation is a central challenge in sequential decision making. Thompson sampling [1] and its extension to reinforcement learning, Posterior Sampling [2], provide an elegant approach that tackles the exploration-exploitation dilemma by maintaining a posterior over models and choosing actions in proportion to the probability that they are optimal. Unfortunately, maintaining such a posterior is intractable in all but the simplest problems.

Several recent papers [3, 4, 5] have explored the idea of using stochastic value functions parameterized by neural networks to choose actions, which might be understood as implementing an approximate form of posterior sampling. Although they showed improvement over naïve exploration strategies (i.e., ϵ-greedy) on the challenging Atari benchmark, we still lack a clear understanding of the strengths and weaknesses of these approaches. Apart from that work, recent developments in Bayesian neural networks [6, 7] provide scalable approaches to approximating the posterior over neural networks. These directly lead to approximate Thompson sampling strategies. In this work, we take a step back and investigate how approximate model posteriors affect the performance of decision making when used with Thompson Sampling in the much simpler contextual bandit problem. This allows us to disentangle the exploration-exploitation dilemma from the challenge of using function approximators with Q-learning.

In the contextual bandit setting, we observe a context $X_t \in \mathbf{R}^d$ at each timestep t, then choose one of the k available actions, a_t, according to an algorithm, and finally receive a reward $r_t = r_t(X_t, a_t)$. The cumulative reward for the algorithm is given by $r = \sum_t r_t$, and cumulative regret is defined as $\mathbf{E}[r^* - r]$, where r^* is the cumulative reward of the optimal policy. The goal is to minimize cumulative regret.

2 Algorithms

We describe the algorithms used to approximately sample from the model posterior in our experiments.

Linear Methods We apply closed-form updates for the posterior in Bayesian linear regression [8] ($r_t = x_t^T \beta + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$) which admit a computationally-efficient online version. We consider exact linear posteriors as a strong baseline. Importantly, we model the *joint* distribution of β and σ^2 for each action. Sequentially estimating the noise level σ^2 for each action allows the

Second workshop on Bayesian Deep Learning (NIPS 2017), Long Beach, CA, USA.

algorithm to adaptively improve its understanding of the volume of the hyperellipsoid of plausible β's, leading, in general, to a more aggressive initial exploration phase (in both β and σ^2).

Neural Linear The main problem linear algorithms face is their lack of representational power, which they complement with accurate uncertainty estimates. A natural attempt at getting the best of both worlds consists in performing a Bayesian linear regression on top of the representation of the last layer of a neural network, similarly to [9].

Neural Greedy We refer to the algorithm that simply trains a neural network and acts greedily (i.e., takes the action whose predicted score for the current context is highest) as **RMS**, as we train it using the RMSProp optimizer. This is our non-linear baseline. We also tried the ϵ-greedy version of the algorithm, where a random action was selected with probability ϵ for some decaying schedule of ϵ.

Variational Inference Variational approaches approximate the posterior by finding a distribution within a tractable family that minimizes the KL divergence to the posterior [10]. Typically (and in our experiments), the posterior is approximated by a mean-field or factorized distribution. Recent advances have scaled these approaches to estimate the posterior of neural networks with millions of parameters [6]. A common criticism of variational inference is that it underestimates uncertainty (e.g., [8]), which could lead to under-exploration.

Dropout [11] can also be seen as optimizing a variational objective [12, 13], which leads to a straight-forward posterior approximation and application to Thompson sampling.

Markov Chain Monte Carlo methods construct a Markov chain whose stationary distribution is the posterior distribution. The Stochastic Gradient Langevin Dynamics (SGLD) methods [14, 15] add Gaussian noise to the model gradients during stochastic gradient updates in such a manner that each update results in an approximate sample from the posterior. Stochastic Gradient Fisher Scoring (SGFS) [16, 17] uses the Fisher information matrix as a preconditioner in SGLD. [18] develops methods for approximately sampling from the posterior using a constant learning rate in SGD, and a prescription for a stable version of SGFS. We evaluate the diagonal-SGFS and constant-SGD algorithms from [18] in this work.

Bootstrap A simple approach to approximate the sampling distribution of any estimator is the Bootstrap [19]. We simultaneously train q models, where each model i is based on a different dataset D_i. If all the data D is available in advance, D_i is typically created by sampling $|D|$ elements from D at random with replacement. In our case, however, the data grows one example at a time. Accordingly, we set a parameter $p \in (0, 1]$, and append the new datapoint to each D_i independently at random with probability p. In order to emulate Thompson Sampling, we sample a model uniformly at random and take the action predicted to be best by the sampled model.

Direct Noise Injection Parameter-Noise [20] is a recently proposed approach for exploration in deep RL that has shown promising results. The training updates for the network are unchanged, but when selecting actions, the network weights are perturbed with isotropic Gaussian noise. Crucially, the network uses layer normalization [21], which ensures that all weights are on the same scale.

Gaussian processes [22] are a gold-standard method for modeling distributions over non-linear continuous functions, and a natural baseline. Our implementation is a multi-task Gaussian process [23] with a linear and Matern $3/2$ product kernel over the inputs and an exponentiated quadratic kernel over latent vectors for the different tasks.

As with any empirical study, there is a limit to the number of experiments that can be run. Although promising, we did not include expectation propagation [24] and α-divergence minimization of which variational inference is a special case [25]. We leave evaluating these algorithms as future work.

3 Empirical Evaluation

We evaluated the algorithms (see Appendix A) on several bandit problems constructed from real-world datasets ([26, 27, 28] and Appendix B for details).

Neural Network Architectures. All algorithms based on neural networks as function approximators used a fully-connected feedforward network with two hidden layers with 100 units each and softplus activations (ReLu activations produced similar results). The network output a value for each of the k outputs.

Updating Models. Ideally, we would train models after every observation until convergence. However, this limits the applicability of our algorithms in online scenarios where decisions must be made

Table 1: Cumulative regret (relative to uniform randomly sampling actions) incurred by algorithms in Section 2 (see Appendix A for details). We report the mean and standard error of the mean over 50 trials.

	Mushroom	Statlog	Covertype	Financial	Jester
BBBN	5.08 ± 1.00	25.23 ± 0.00	59.89 ± 0.05	41.24 ± 2.17	68.04 ± 0.81
BBBN2	4.16 ± 1.04	25.23 ± 0.00	60.38 ± 0.31	50.04 ± 3.52	66.23 ± 0.86
BBBN3	9.42 ± 2.55	25.23 ± 0.00	60.21 ± 0.29	55.32 ± 3.70	65.02 ± 0.84
Bootstrapped NN	3.99 ± 0.20	1.58 ± 0.05	32.03 ± 0.89	14.76 ± 0.63	75.15 ± 0.54
Bootstrapped NN2	2.25 ± 0.09	1.73 ± 0.07	32.09 ± 0.91	17.67 ± 1.13	74.63 ± 0.63
Dropout (RMS3)	1.89 ± 0.08	8.48 ± 0.79	33.34 ± 0.89	16.82 ± 0.80	66.00 ± 0.56
Dropout (RMS2)	$\mathbf{1.70 \pm 0.06}$	7.35 ± 0.72	33.13 ± 0.78	16.09 ± 0.73	66.17 ± 0.75
GP	16.67 ± 0.79	2.65 ± 0.47	40.77 ± 0.31	4.23 ± 0.07	75.02 ± 0.83
Neural Linear	2.27 ± 0.10	1.15 ± 0.01	26.24 ± 0.05	6.90 ± 0.08	73.85 ± 0.46
RMS1	5.98 ± 0.40	2.49 ± 0.22	28.27 ± 0.39	15.96 ± 0.54	72.89 ± 0.64
RMS2	$\mathbf{1.84 \pm 0.09}$	3.32 ± 0.55	36.93 ± 1.55	20.00 ± 1.14	71.84 ± 0.68
RMS2b	3.42 ± 1.04	1.97 ± 0.46	29.58 ± 0.83	7.56 ± 0.62	73.24 ± 0.81
RMS3	1.94 ± 0.13	2.65 ± 0.29	36.02 ± 1.58	17.15 ± 0.91	71.05 ± 0.75
SGFS	3.80 ± 0.18	2.83 ± 0.27	36.48 ± 0.12	23.37 ± 0.74	69.52 ± 0.65
ConstSGD	7.38 ± 1.69	$\mathbf{0.86 \pm 0.13}$	$\mathbf{21.90 \pm 0.18}$	53.85 ± 3.28	73.62 ± 0.72
EpsGreedy (RMS1)	7.15 ± 0.33	2.32 ± 0.11	27.27 ± 0.16	15.33 ± 0.52	74.37 ± 0.59
EpsGreedy (RMS2)	2.29 ± 0.10	2.20 ± 0.14	31.31 ± 0.23	17.85 ± 0.94	71.42 ± 0.78
EpsGreedy (RMS3)	2.31 ± 0.11	2.24 ± 0.14	31.21 ± 0.21	16.78 ± 0.66	71.94 ± 0.77
LinDiagPost	17.78 ± 0.23	51.26 ± 0.03	95.49 ± 0.02	9.28 ± 0.07	$\mathbf{59.11 \pm 0.47}$
LinDiagPrecPost	9.66 ± 1.31	7.52 ± 0.02	34.41 ± 0.02	4.49 ± 0.05	$\mathbf{58.15 \pm 0.56}$
LinGreedy	14.16 ± 1.66	12.76 ± 0.67	35.22 ± 0.23	$\mathbf{2.71 \pm 0.35}$	$\mathbf{58.52 \pm 0.39}$
LinPost	6.11 ± 0.71	7.64 ± 0.02	34.38 ± 0.02	7.02 ± 0.06	$\mathbf{58.25 \pm 0.48}$
LinfullDiagPost	86.73 ± 0.11	28.29 ± 0.02	73.76 ± 0.03	6.82 ± 0.07	63.59 ± 0.49
LinfullDiagPrecPost	5.24 ± 0.77	7.37 ± 0.02	34.03 ± 0.03	3.89 ± 0.04	61.21 ± 0.49
LinfullPost	2.75 ± 0.31	7.36 ± 0.02	34.01 ± 0.02	5.40 ± 0.05	60.85 ± 0.51
Param-Noise	2.13 ± 0.16	2.13 ± 0.23	33.25 ± 0.75	16.96 ± 0.90	71.25 ± 0.77
Param-Noise2	4.67 ± 1.44	1.32 ± 0.23	29.01 ± 0.26	7.70 ± 0.52	73.57 ± 0.67
Uniform	100.00 ± 0.18	100.00 ± 0.03	100.00 ± 0.01	100.00 ± 1.60	100.00 ± 1.24

immediately. We trained the neural networks for 20 mini-batches every 20 timesteps[1]. The size of each mini-batch was 512. We experimented with training on more mini-batches every training iteration and found that it was essential for some algorithms like the variational inference approaches.

Hyper-Parameter Tuning In the bandit scenario, we do not have access to each problem a-priori for tuning. Thus, for each algorithm, the hyperparameters were tuned once and shared across all tasks.

We evaluated the algorithms on a range of bandit problems created from real-world data. Briefly, from classification datasets Mushroom, Statlog, Covertype, Financial, and Jester datasets [31], we create bandit problems with 0/1 rewards as is standard (see Appendix A for details). They exhibit a broad range of properties: small and large sizes, one dominating action versus more homogeneous optimality, learnable or little signal, stochastic or deterministic rewards, etc. We summarize the final cumulative regret in Table 1.

4 Discussion

No single algorithm bested the others in every bandit problem, however, we observed some general trends. We found that although bootstrapping, dropout, and injecting random noise helped for some problems, these strategies did not significantly and systematically improve performance in these problems. This suggests that the intrinsic randomness of stochastic gradient descent is enough to explore in many cases. Other algorithms, like Variational Inference and Monte Carlo approaches, strongly couple their complex representation and uncertainty estimates. This proves problematic when decisions are made based on partial optimization of both, as online scenarios usually require. On the other hand, making decisions according to a Bayesian linear regression on the representation provided by the last layer of a deep network offers a robust and easy-to-tune approach. It would be interesting to try this approach on more complex RL domains.

Finally, a limitation of the study is that we assume that if we had access to the actual posterior at all times t, then choosing actions using Thompson Sampling would lead to near-optimal cumulative regret or, more informally, to good performance. In some problems, this is not the case; for example, when actions that have no chance of being optimal still convey useful information about other actions. Thompson Sampling (and UCB approaches) would never select such actions, even if they are worth their cost [32]. In addition, Thompson Sampling does *not* take into account the time horizon where the process ends, and if known, exploration efforts could be tuned accordingly [33].

[1]For comparison, Deep Q-Networks trained on Atari games were updated after every 4 actions [29, 3, 20, 30].

References

[1] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[2] Ian Osband, Dan Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, pages 3003–3011, 2013.

[3] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*, pages 4026–4034, 2016.

[4] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

[5] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

[6] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622, 2015.

[7] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

[8] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[9] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, 2015.

[10] Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Computational learning theory*, pages 5–13. ACM, 1993.

[11] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[12] Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.

[13] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International conference on machine learning*, pages 1050–1059, 2016.

[14] Radford M. Neal. Bayesian learning for neural networks. *Dept. of Computer Science, University of Toronto*, 1994.

[15] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *International Conference on Machine Learning*, 2011.

[16] Sam Patterson and Yee Whye Teh. Stochastic gradient riemannian langevin dynamics on the probability simplex. In *Advances in Neural Information Processing Systems*. 2013.

[17] Sungjin Ahn, Anoop Korattikara Balan, and Max Welling. Bayesian posterior sampling via stochastic gradient fisher scoring. In *International Conference on Machine Learning*, 2012.

[18] Stephan Mandt, Matthew D. Hoffman, and David M. Blei. A variational analysis of stochastic gradient algorithms. In *International Conference on Machine Learning*, 2016.

[19] Bradley Efron. *The jackknife, the bootstrap and other resampling plans*. SIAM, 1982.

[20] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y. Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv:1706.01905*, 2017.

[21] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[22] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[23] Edwin V Bonilla, Kian M. Chai, and Christopher Williams. Multi-task gaussian process prediction. In *Advances in Neural Information Processing Systems*. 2008.

[24] José Miguel Hernández-Lobato and Ryan P. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, 2015.

[25] José Miguel Hernández-Lobato, Yingzhen Li, Mark Rowland, Thang D. Bui, Daniel Hernández-Lobato, and Richard E. Turner. Black-box alpha divergence minimization. In *International Conference on Machine Learning*, 2016.

[26] Robin Allesiardo, Raphaël Féraud, and Djallel Bouneffouf. A neural networks committee for the contextual bandit problem. In *International Conference on Neural Information Processing*, pages 374–381. Springer, 2014.

[27] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning*, pages 1638–1646, 2014.

[28] Raphaël Féraud, Robin Allesiardo, Tanguy Urvoy, and Fabrice Clérot. Random forest for the contextual bandit problem. In *Artificial Intelligence and Statistics*, pages 93–101, 2016.

[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[30] Bilal Piot Jacob Menick Ian Osband Alex Graves Vlad Mnih Remi Munos Demis Hassabis Olivier Pietquin Charles Blundell Shane Legg Meire Fortunato, Mohammad Gheshlaghi Azar. Noisy networks for exploration. *arXiv:1706.10295*, 2017.

[31] Arthur Asuncion and David Newman. UCI machine learning repository, 2007.

[32] Dan Russo and Benjamin Van Roy. Learning to optimize via information-directed sampling. In *Advances in Neural Information Processing Systems*, pages 1583–1591, 2014.

[33] Daniel Russo, David Tse, and Benjamin Van Roy. Time-sensitive bandit learning and satisficing thompson sampling. *arXiv preprint arXiv:1704.09028*, 2017.

[34] Jeff Schlimmer. Mushroom records drawn from the audubon society field guide to north american mushrooms. *GH Lincoff (Pres), New York*, 1981.

[35] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[36] Carlos Riquelme, Mohammad Ghavamzadeh, and Alessandro Lazaric. Active learning for accurate estimation of linear models. In *International Conference on Machine Learning*, 2017.

[37] Adam N Elmachtoub, Ryan McNellis, Sechan Oh, and Marek Petrik. A practical method for solving contextual bandit problems using decision trees. *arXiv preprint arXiv:1706.04687*, 2017.

Appendix

A Algorithms

Table 2: Detailed description of the algorithms in the experiments. Unless otherwise stated, algorithms use $t_s = 20$ (mini-batches per training period), and $t_f = 20$ (one training period every t_f contexts).

Algorithm	Description
BBBN	BayesByBackprop with $\sigma = 0.5$. ($t_s = 500$, first 15 times $t_s = 10000$).
BBBN2	BayesByBackprop with $\sigma = 0.75$. ($t_s = 500$, first 15 times $t_s = 10000$).
BBBN3	BayesByBackprop with $\sigma = 1.0$. ($t_s = 500$, first 15 times $t_s = 10000$).
Bootstrapped NN	Bootstrapped with $q = 5$ models, and $p = 0.85$. Based on RMS3 net.
Bootstrapped NN2	Bootstrapped with $q = 5$ models, and $p = 1.0$. Based on RMS3 net.
Dropout (RMS3)	Dropout with probability $p = 0.8$. Based on RMS3 net.
Dropout (RMS2)	Dropout with probability $p = 0.8$. Based on RMS2 net.
GP	For computational reasons, it only uses the first 1000 data points.
Neural Linear	Noise prior $a_0 = 6$, $b_0 = 6$. Ridge prior $\lambda = 0.25$. Based on RMS2 net.
RMS1	Greedy NN approach, fixed learning rate ($\gamma = 0.01$).
RMS2	Learning rate decays, and it is reset every training period.
RMS2b	Similar to RMS2, but training for longer ($t_s = 500$).
RMS3	Learning rate decays, and it is not reset at all. Starts at $\gamma = 1$.
SGFS	Burning $= 500$, learning rate $\gamma = 0.014$, EMA decay $= 0.9$, noise $\sigma = 0.75$.
ConstSGD	Burning $= 500$, EMA decay $= 0.9$, noise $\sigma = 0.5$.
EpsGreedy (RMS1)	Initial $\epsilon = 0.01$. Multiplied by 0.999 after every context. Based on RMS1 net.
EpsGreedy (RMS2)	Initial $\epsilon = 0.01$. Multiplied by 0.999 after every context. Based on RMS2 net.
EpsGreedy (RMS3)	Initial $\epsilon = 0.01$. Multiplied by 0.999 after every context. Based on RMS3 net.
LinDiagPost	Σ in Eq. **??** is diagonalized. Ridge prior $\lambda = 0.25$. Assumed noise level $\sigma^2 = 0.25$.
LinDiagPrecPost	Σ^{-1} in Eq. **??** is diagonalized. Ridge prior $\lambda = 0.25$. Assumed noise level $\sigma^2 = 0.25$.
LinGreedy	Ridge prior $\lambda = 0.25$. Assumed noise level $\sigma^2 = 0.25$.
LinPost	Ridge prior $\lambda = 0.25$. Assumed noise level $\sigma^2 = 0.25$.
LinfullDiagPost	Σ in Eq. **??** is diagonalized. Noise prior $a_0 = 6$, $b_0 = 6$. Ridge prior $\lambda = 0.25$.
LinfullDiagPrecPost	Σ^{-1} in Eq. **??** is diagonalized. Noise prior $a_0 = 6$, $b_0 = 6$. Ridge prior $\lambda = 0.25$.
LinfullPost	Noise prior $a_0 = 6$, $b_0 = 6$. Ridge prior $\lambda = 0.25$.
Param-Noise	Initial noise $\sigma = 0.01$, and level $\epsilon = 0.01$. Based on RMS3 net.
Param-Noise2	Initial noise $\sigma = 0.01$, and level $\epsilon = 0.01$. Based on RMS3 net. Trained for longer: $t_s = 800$.
Uniform	Takes each action at random with equal probability.

B Real-World Datasets

Mushroom. The Mushroom Dataset [34] contains 22 attributes per mushroom, and two classes: poisonous and safe. As in [6], we create a bandit problem where the agent must decide whether to eat or not a given mushroom. Eating a safe mushroom provides reward +5. Eating a poisonous mushroom delivers reward +5 with probability 1/2 and reward -35 otherwise. If the agent does not eat a mushroom, then the reward is 0. We set $n = 50000$.

Statlog. The Shuttle Statlog Dataset [31] provides the value of $d = 9$ indicators during a space shuttle flight, and the goal is to predict the state of the radiator subsystem of the shuttle. There are $k = 7$ possible states, and if the agent selects the right state, then reward 1 is generated. Otherwise, the agent obtains no reward ($r = 0$). The most interesting aspect of the dataset is that one action is the optimal one in 80% of the cases, and some algorithms may commit to this action instead of further exploring. In this case, $n = 43500$.

Covertype. The Covertype Dataset [31] classifies the cover type of northern Colorado forest areas in $k = 7$ classes, based on $d = 54$ features, including elevation, slope, aspect, and soil type. Again, the agent obtains reward 1 if the correct class is selected, and 0 otherwise. We run the bandit for $n = 150000$.

Financial. We created the Financial Dataset by pulling the stock prices of $d = 21$ publicly traded companies in NYSE and Nasdaq, for the last 14 years ($n = 3713$). For each day, the context was the price difference between the beginning and end of the session for each stock. We synthetically created the arms, to be a linear combination of the contexts, representing $k = 8$ different potential portfolios. By far, this was the smallest dataset, and many algorithms over-explored at the beginning with no time to amortize their investment (Thompson Sampling does not account for the horizon).

Jester. We create a recommendation system bandit problem as follows. The Jester Dataset [35] provides continuous ratings in $[-10, 10]$ for 100 jokes from 73421 users. We find a complete subset of $n = 19181$ users rating all 40 jokes. Following [36], we take $d = 32$ of the ratings as the context of the user, and $k = 8$ as the arms. The agent recommends one joke, and obtains the reward corresponding to the rating of the user for the selected joke.

The Statlog and Covertype datasets were tested in [37].