Meta-Learning Contextual Bandit Exploration

Amr Sharaf
University of Maryland
amr@cs.umd.edu

Hal Daumé III
Microsoft Research &
University of Maryland
me@hal3.name

Abstract

In contextual bandits, an algorithm must choose actions given observed contexts, learning from a reward signal that is observed only for the action chosen. This leads to an exploration/exploitation trade-off: the algorithm must balance taking actions it already believes are good with taking new actions to potentially discover better choices. We develop a meta-learning algorithm, MÊLÉE, that learns an exploration policy based on simulated, synthetic contextual bandit tasks. MÊLÉE learns via imitation learning against these simulations to learn an exploration policy that can be applied to true contextual bandit tasks at test time. In hundreds of real-world datasets, MÊLÉE outperforms seven strong baseline algorithms on most datasets, and can leverage a rich feature representation for learning an exploration strategy.

1 Introduction

In a contextual bandit problem, an agent optimizes its behavior over a sequence of rounds based on limited feedback (Kaelbling, 1994; Auer, 2003; Langford & Zhang, 2008). In each round, the agent chooses an action based on a context (features) for that round, and observes a reward for that action but no others. Contextual bandit problems arise in many real-world settings like online recommendations and personalized medicine. As in reinforcement learning, the agent must learn to balance *exploitation* and *exploration*.

We present a meta-learning approach to automatically learn a good exploration mechanism from data—specifically, simulated, synthetic offline data. Based on these simulations, our algorithm, MÊLÉE (MEta LEarner for Exploration)[1], learns a good heuristic exploration strategy that should ideally generalize to future contextual bandit problems. At training time (§2), MÊLÉE simulates many contextual bandit problems from fully labeled synthetic data. Thus, MÊLÉE is able to counterfactually simulate what would happen under all possible action choices. It then uses this information to compute regret estimates for each action, which can be optimized using the AggreVaTe imitation learning algorithm (Ross & Bagnell, 2014). Our imitation learning strategy mirrors that of the meta-learning approach of Bachman et al. (2017) in the active learning setting. Empirically, we use MÊLÉE to train an exploration policy on only synthetic datasets and evaluate the resulting bandit performance across three hundred (simulated) contextual bandit tasks (§3), comparing to a number of alternative exploration algorithms, and showing the efficacy of our approach (§3).

2 Approach: Learning and Effective Exploration Strategy

In order to have an effective approach to the contextual bandit problem, one must be able to both optimize a policy based on historic data and make decisions about how to explore. The exploration/exploitation dilemma is fundamentally about long-term payoffs: is it worth trying something potentially suboptimal *now* in order to learn how to behave better in the future? A particularly simple and effective form of exploration is ϵ-greedy: given a function f output by POLOPT, act according to $f(x)$ with probability $(1-\epsilon)$ and act uniformly at random with probability ϵ. Intuitively, one would hope to improve on such a strategy by taking more (any!) information into account; for instance,

[1] **Code:** https://www.dropbox.com/sh/dc3v8po5cbu8zaw/AACu1f_4c4wIZxD1e7W0KVZ0a?dl=0

3rd Workshop on Meta-Learning at NeurIPS 2019, Vancouver, Canada

Algorithm 1 MÊLÉE (supervised training sets $\{S_m\}$, hypothesis class $\mathcal{F}$, exploration rate μ, number of validation examples N_{Val}, feature extractor Φ)

for round $n = 1, 2, \ldots, N$ **do**
  initialize meta-dataset $D = \{\}$, choose S at random from $\{S_m\}$, and set history $h_0 = \{\}$
  partition and permute S randomly into train *Tr* and validation *Val* where $|Val| = N_{Val}$
  for round $t = 1, 2, \ldots, |Tr|$, and $(x_t, \boldsymbol{r}_t) = Tr_t$ **do**
    for each action $a = 1, \ldots, K$ **do**
      optimize $f_{t,a} = \text{POLOPT}(\mathcal{F}, h_{t-1} \oplus (x_t, a, r_t(a), 1 - \frac{K-1}{K}\mu))$ on augmented history
      roll-out: estimate $\hat{c}_a$, the cost-to-go of a, using $r_t(a)$ and a roll-out policy π^{out} on $f_{t,a}$
    end for
    compute $f_t = \text{POLOPT}(\mathcal{F}, h_{t-1})$
    aggregate $D \leftarrow D \oplus (\Phi(f_t, x_t, h_{t-1}, Val), \langle \hat{c}_1, \ldots, \hat{c}_K \rangle)$
    roll-in: $a_t \sim \frac{\mu}{K} 1_K + (1 - \mu)\pi_{n-1}(f_t, x_t)$ with probability p_t, where 1 is the ones-vector
    append history $h_t \leftarrow h_{t-1} \oplus (x_t, a_t, r_t(a_t), p_t)$
  end for
  update $\pi_n = \text{LEARN}(D)$
end for
return $\{\pi_n\}_{n=1}^N$

basing the probability of exploration on f's uncertainty. In this section, we describe MÊLÉE, first by describing how it operates at test time when applied to a new contextual bandit problem (§2), and then by describing how to train it using synthetic simulated contextual bandit problems (§2).

Test Time Behavior of MÊLÉE. Our goal in this paper is to *learn* how to explore from experience. The training procedure for MÊLÉE will use offline supervised learning problems to learn an *exploration policy* π, which takes *two inputs*: a function $f \in \mathcal{F}$ and a context x, and outputs an action. This is used to produce an agent which interacts with the world, maintaining an initially empty history buffer h, as: 1. The world draws $(x_t, \boldsymbol{r}_t) \sim \mathcal{D}$ and reveals context x_t. 2. The agent computes $f_t \leftarrow \text{POLOPT}(h)$ and a greedy action $\tilde{a}_t = \pi(f_t, x_t)$. 3. The agent plays $a_t = \tilde{a}_t$ with probability $(1 - \mu)$, and a_t uniformly at random otherwise. 4. The agent observes $r_t(a_t)$ and appends $(x_t, a_t, r_t(a_t), p_t)$ to the history h, where $p_t = \mu/K$ if $a_t \neq \tilde{a}_t$; and $p_t = 1 - \mu + \mu/K$ if $a_t = \tilde{a}_t$. Here, f_t is the function optimized on the historical data, and π uses it and x_t to choose an action. Intuitively, π might choose to use the prediction $f_t(x_t)$ most of the time, unless f_t is quite uncertain on this example, in which case π might choose to return the second (or third) most likely action according to f_t. The agent then performs a small amount of additional μ-greedy-style exploration: most of the time it acts according to π but occasionally it explores some more. In practice (§3), we find that setting $\mu = 0$ is optimal in aggregate, but non-zero μ is necessary for our theory (Appendix A).

Training MÊLÉE by Imitation Learning. The meta-learning challenge is: how do we learn a good exploration policy π? We assume access to *fully labeled* data on which we can train π; this data must include context/reward pairs, but where the reward for *all* actions is known. This is a weak assumption: in practice, we use purely synthetic data; one could alternatively use any fully labeled classification dataset (Beygelzimer & Langford, 2009). Under this assumption, π's behavior is a sequential decision making problem in a simulated setting, for which imitation learning (Daumé et al., 2009; Ross et al., 2011; Ross & Bagnell, 2014; Chang et al., 2015) is a natural solution.[2]

At training time, MÊLÉE will treat one of these synthetic datasets as if it were a contextual bandit dataset. At each step t, it computes f_t by running POLOPT on the historical data, and then asks: for *each* action, what would the long time reward look like if I were to take this action (which can be evaluated for each possible action), and a policy π can be learned to maximize these rewards. More formally, in imitation learning, we assume training-time access to an *expert*, $\pi^\star$, whose behavior we wish to learn to imitate at test-time. From this, we can define an optimal reference policy $\pi^\star$, which effectively "cheats" at training time by looking at the true labels. The learning problem is then to estimate π to have as similar behavior to $\pi^\star$ as possible, but without access to those labels.

The imitation learning algorithm we use is AggreVaTe (Ross & Bagnell, 2014) (closely related to DAgger (Ross et al., 2011)), and is instantiated for the contextual bandits meta-learning problem in

[2] In other work on meta-learning, such problems are often cast as full *reinforcement-learning* problems. We opt for imitation learning instead because it is computationally attractive and effective when a simulator exists.

Alg 1. AggreVaTe learns to choose actions to minimize the cost-to-go of the expert rather than the zero-one classification loss of mimicking its actions. Following the AggreVaTe template, MÊLÉE operates in an iterative fashion, starting with an arbitrary π and improving it through interaction with an expert. Over N rounds, MÊLÉE selects random training sets and simulates the test-time behavior on that training set. The core functionality is to generate a number of states (f_t, x_t) on which to train π, and to use the supervised data to estimate the value of every action from those states. MÊLÉE achieves this by sampling a random supervised training set and setting aside some validation data from it (line 3). It then simulates a contextual bandit problem on this training data; at each time step t, it tries *all* actions and "pretends" like they were appended to the current history (line 6) on which it trains a new policy and evaluates it's **roll-out value** (line 7). This yields, for each t, a new training example for π, which is added to π's training set (line 10); the features for this example are features of the classifier based on true history (line 9) (and possibly statistics of the history itself), with a label that gives, for each action, the corresponding cost-to-go of that action (the c_as computed in line 7). MÊLÉE then must commit to a **roll-in action** to *actually* take; it chooses this in line 11.

3 Experimental Setup and Results

Using a collection of synthetically generated classification problems, we train an exploration policy π using MÊLÉE (Alg 1). This exploration policy learns to explore on the basis of calibrated probabilistic predictions from f together with a predefined set of exploration features (§B.1). Once π is learned and fixed, we follow the test-time behavior described in §2 on a set of 300 "simulated" contextual bandit problems, derived from standard classification tasks. In all cases, the underlying classifier f is a linear model trained with a policy optimizer that runs stochastic gradient descent (details are in §B.3). We seek to answer two questions experimentally: (1) How does MÊLÉE compare empirically to alternative (expert designed) exploration strategies? (2) How important are the additional features used by MÊLÉE in comparison to using calibrated probability predictions from f as features?

Evaluation Tasks. Following Bietti et al. (2018), we use a collection of 300 binary classification datasets from `openml.org` for evaluation; We convert classification datasets into cost-sensitive classification problems by using a $0/1$ encoding.Given these fully supervised datasets, we simulate the contextual bandit setting by only revealing the reward for the selected actions.

Evaluation Metrics. For evaluation, we use progressive validation (Blum et al., 1999), which is exactly computing the reward of the algorithm. Specifically, to evaluate the performance of an exploration algorithm $\mathcal{A}$ on a dataset S of size n, we compute the progressive validation return $G(\mathcal{A}) = \frac{1}{n}\sum_{t=1}^{n} r_t(a_t)$ as the average reward up to n, where a_t is the action chosen by the algorithm $\mathcal{A}$ and r_t is the true reward vector. Because our evaluation is over 300 datasets, we report aggregate results in two forms. The simpler one is **Win/Loss Statistics:** We compare two exploration methods on a given dataset by counting the number of statistically significant wins and losses. An exploration algorithm $\mathcal{A}$ wins over another algorithm $\mathcal{B}$ if the progressive validation return $G(\mathcal{A})$ is statistically significantly larger than B's return $G(\mathcal{B})$ at the 0.01 level using a paired sample t-test. We also report **cumulative distributions** of rewards for each algorithm, following Zhang et al. (2019). In particular, for a given relative reward value ($x \in [0, 1]$), the corresponding CDF value for a given algorithm is the fraction of datasets on which this algorithm achieved reward at least x. We compute relative reward by Min-Max normalization: linearly transforming reward y to $y' = \frac{y-\min}{\max-\min}$, where min & max are the minimum & maximum rewards among all exploration algorithms.

Baselines. In our experiments, we compare to the following baseline exploration methods, keeping the policy optimization method fixed (details in §B.4): **ϵ-greedy** (Sutton, 1996); **ϵ-decreasing** (Sutton & Barto, 1998); **EG ϵ-greedy** (Li et al., 2010b); **τ-first**. We additionally compare to three state-of-the-art exploration methods: **LinUCB** (Li et al., 2010a); **Cover** (Agarwal et al., 2014); and its variant **Cover-NU** (Bietti et al., 2018). We select the best hyperparameters following Bietti et al. (2018).

Experimental Results. In Figure 1 (left), we show a representative learning curve. Here, we see that as more data becomes available, all the approaches improve (except τ-first, which has ceased to learn after 2% of the data). MÊLÉE, in particular, is able to very quickly achieve near-optimal performance (in around 40 examples) in comparison to the best baseline which takes at least 200. In Figure 1 (middle), we show the CDFs for the different algorithms. To help read this, at $x = 1.0$, MÊLÉE has a relative reward at least 1.0 on more than 40% of datasets, while ϵ-decreasing and

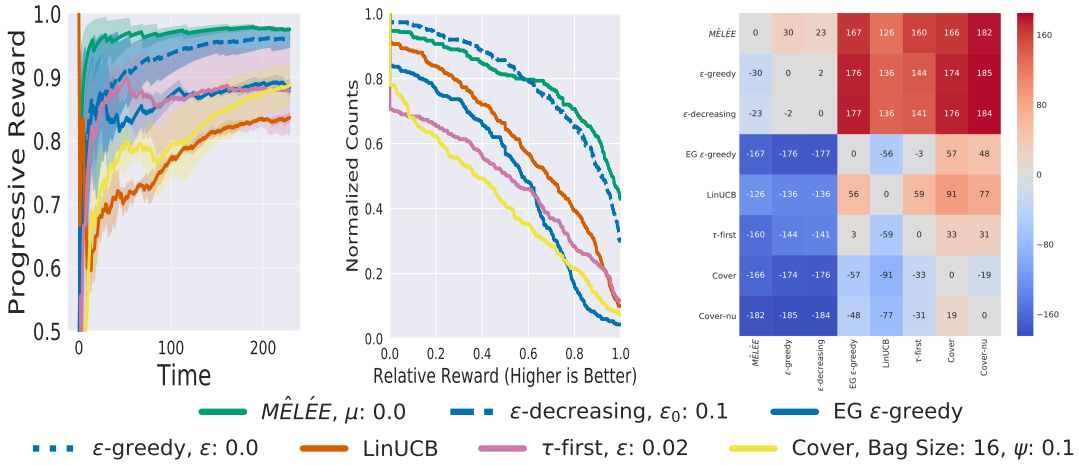


Figure 1: Behavior of MÊLÉE in comparison to baseline and state-of-the-art exploration algorithms. **(Left)** A representative learning curve on dataset #1144. **(Middle)** Comparison of all exploration algorithms using the empirical cumulative distribution function of the relative progressive validation return G (upper-right is optimal) on 300 classification problems. The CDF curves for ϵ-decreasing and ϵ-greedy coincide. **(Right)** Win/Loss counts for all algorithms.

ϵ-greedy achieve this on about 30% of datasets. We find that the two strongest baselines are ϵ-decreasing and ϵ-greedy (better when reward differences are small, toward the left of the graph). The two curves for ϵ-decreasing and ϵ-greedy coincide. This happens because the exploration probability ϵ_0 for ϵ-decreasing decays rapidly approaching zero with a rate of $\frac{1}{t}$, where t is the index of the current round. MÊLÉE outperforms the baselines in the "large reward" regimes (right of graph) but under-performs ϵ-decreasing and ϵ-greedy in low reward regimes (left of graph). In Figure 1 (right), we show statistically-significant win/loss differences for each of the algorithms. Here, each (row, column) entry shows the number of times the row algorithm won against the column, minus the number of losses. MÊLÉE is the only algorithm that always wins more than it loses against other algorithms, and outperforms the nearest competition (ϵ-decreasing) by 23 points.

4 Discussion

Meta-learning has been studied in the context of active learning, including reinforcement learning-based strategies (Woodward & Finn, 2017; Fang et al., 2017), imitation learning-based strategies (Bachman et al., 2017), and batch supervised learning-based strategies (Konyushkova et al., 2017). While meta-learning for contextual bandits is most similar to meta-learning for active learning, there is a fundamental difference that makes it significantly more challenging: in active learning, the goal is to select as few examples as you can to learn, so by definition the horizon is short; in contextual bandits, learning to explore is fundamentally a long-horizon problem, because what matters is not immediate reward but long term learning. In reinforcement learning, Gupta et al. (2018) and Xu et al. (2018) investigated the task of meta-learning an exploration strategy for a distribution of related tasks by learning a latent exploration space. While these approaches are effective if the distribution of tasks is very similar and the state space is shared among different tasks, they fail to generalize when the tasks are different. Our approach targets an easier problem than exploration in full reinforcement learning environments, and can generalize well across a wide range of different tasks with completely unrelated features spaces. There has also been a substantial amount of work on constructing "good" exploration policies, in problems of varying complexity: traditional bandit settings (Karnin & Anava, 2016), contextual bandits (Fraud et al., 2016) and reinforcement learning (Osband et al., 2016). MÊLÉE, outperforms alternative exploration algorithm in most settings, especially when reward differences are large. One limitation of MÊLÉE is the computational resources required during the offline training phase on the synthetic datasets. Despite fairly substantial distribution mismatch (synthetic $\rightarrow$ real-world), MÊLÉE works well in practice, and our stylized theory (Appendix A) suggests that there may be an interesting avenue for developing strong theoretical results for contextual bandit learning with learned exploration policies, and perhaps other meta-learning problems.

Acknowledgments

We thank members of the CLIP lab for reviewing earlier versions of this work. This material is based upon work supported by the National Science Foundation under Grant No. 1618193. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Agarwal, A., Hsu, D., Kale, S., Langford, J., Li, L., and Schapire, R. E. Taming the monster: A fast and simple algorithm for contextual bandits. In *In Proceedings of the 31st International Conference on Machine Learning (ICML-14*, pp. 1638–1646, 2014.

Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, 3:397–422, 2003.

Bachman, P., Sordoni, A., and Trischler, A. Learning algorithms for active learning. In *ICML*, 2017.

Beygelzimer, A. and Langford, J. The offset tree for learning with partial labels. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 129–138. ACM, 2009.

Bietti, A., Agarwal, A., and Langford, J. A Contextual Bandit Bake-off. working paper or preprint, May 2018.

Blum, A., Kalai, A., and Langford, J. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Proceedings of the twelfth annual conference on Computational learning theory*, pp. 203–208. ACM, 1999.

Breiman, L. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324.

Chang, K.-W., Krishnamurthy, A., Agarwal, A., Daumé, III, H., and Langford, J. Learning to search better than your teacher. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML, pp. 2058–2066. JMLR.org, 2015.

Daumé, III, H., Langford, J., and Marcu, D. Search-based structured prediction. *Machine Learning*, 75(3):297–325, Jun 2009. ISSN 1573-0565. doi: 10.1007/s10994-009-5106-x.

Fang, M., Li, Y., and Cohn, T. Learning how to active learn: A deep reinforcement learning approach. In *EMNLP*, 2017.

Fraud, R., Allesiardo, R., Urvoy, T., and Clrot, F. Random forest for the contextual bandit problem. In Gretton, A. and Robert, C. C. (eds.), *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pp. 93–101, Cadiz, Spain, 09–11 May 2016. PMLR.

Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., and Levine, S. Meta-reinforcement learning of structured exploration strategies. *arXiv preprint arXiv:1802.07245*, 2018.

Hazan, E. et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Kaelbling, L. P. Associative reinforcement learning: Functions ink-dnf. *Machine Learning*, 15(3): 279–298, 1994.

Kakade, S. M., Shalev-Shwart, S., and Tewari, A. Efficient bandit algorithms for online multiclass prediction. In *ICML*, 2008.

Karnin, Z. S. and Anava, O. Multi-armed bandits: Competing with optimal sequences. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 199–207. Curran Associates, Inc., 2016.

Konyushkova, K., Sznitman, R., and Fua, P. Learning active learning from data. In *Advances in Neural Information Processing Systems*, 2017.

Langford, J. and Zhang, T. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems 20*, pp. 817–824. Curran Associates, Inc., 2008.

Li, L., Chu, W., Langford, J., and Schapire, R. E. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pp. 661–670, New York, NY, USA, 2010a. ACM. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772758.

Li, W., Wang, X., Zhang, R., Cui, Y., Mao, J., and Jin, R. Exploitation and exploration in a performance based contextual advertising system. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pp. 27–36, New York, NY, USA, 2010b. ACM.

Lin, H.-T., Lin, C.-J., and Weng, R. C. A note on platt's probabilistic outputs for support vector machines. *Machine Learning*, 68(3):267–276, Oct 2007. ISSN 1573-0565. doi: 10.1007/s10994-007-5018-6.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 4026–4034. Curran Associates, Inc., 2016.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Peters, J., Mooij, J. M., Janzing, D., and Schölkopf, B. Causal discovery with continuous additive noise models. *The Journal of Machine Learning Research*, 15(1):2009–2053, 2014.

Platt, J. C. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pp. 61–74. MIT Press, 1999.

Ross, S. and Bagnell, J. A. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

Ross, S., Gordon, G., and Bagnell, J. A. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

Sutton, R. S. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in neural information processing systems*, pp. 1038–1044, 1996.

Sutton, R. S. and Barto, A. G. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

Woodward, M. and Finn, C. Active one-shot learning. *arXiv preprint arXiv:1702.06559*, 2017.

Xu, T., Liu, Q., Zhao, L., Xu, W., and Peng, J. Learning to explore with meta-policy gradient. *arXiv preprint arXiv:1803.05044*, 2018.

Zhang, C., Agarwal, A., Daumé, III, H., Langford, J., and Negahban, S. N. Warm-starting contextual bandits: Robustly combining supervised and bandit feedback. In *ICML*, 2019.

Supplementary Material For: Meta-Learning Contextual Bandit Exploration

A Theoretical Guarantees

We analyze MÊLÉE, showing that the no-regret property of AGGREVATE can be leveraged in our meta-learning setting for learning contextual bandit exploration. In particular, we first relate the regret of the learner in line 16 to the overall regret of π. This will show that, *if* the underlying classifier improves sufficiently quickly, MÊLÉE will achieve sublinear regret. We then show that for a specific choice of underlying classifier (BANDITRON), this is achieved. MÊLÉE is an instantiation of AGGREVATE (Ross & Bagnell, 2014); as such, it inherits AGGREVATE's regret guarantees.

Theorem 1 (Thm 2.2 of Ross & Bagnell (2014), adapted) *After N rounds, if* LEARN *(line 14) is no-regret algorithm, then as $N \to \infty$, with probability 1, it holds that $J(\bar{\pi}) \leq J(\pi^\star) + 2T\sqrt{K\hat{\epsilon}_{class}(T)}$, where $J(\cdot)$ is the reward of the exploration policy, $\bar{\pi}$ is the average policy returned, and $\hat{\epsilon}_{class}(T)$ is the average regression regret for each π_n accurately predicting $\hat{c}$, where*

$$\hat{\epsilon}_{class}(T) = \min_{\pi \in \Pi} \frac{1}{N} \hat{\mathbb{E}}_{t \sim U(T), s \sim d^t_{\pi_i}} \sum_{i=1}^{N} \left[Q^\star_{T-t+1}(s, \pi) - \min_a Q^\star_{T-t+1}(s, a) \right] \quad (1)$$

is the empirical minimum expected cost-sensitive classification regret achieved by policies in the class Π on all the data over the N iterations of training when compared to the Bayes optimal regressor, for $U(T)$ the uniform distribution over $\{1, \ldots, T\}$, d^t_π the distribution of states at time t induced by executing policy π, and $Q^\star$ the cost-to-go of the expert.

Thus, achieving low regret at the problem of learning π on the training data it observes ("D" in MÊLÉE), i.e. $\hat{\epsilon}_{class}(T)$ is small, translates into low regret in the contextual-bandit setting. At first glance this bound looks like it may scale linearly with T. However, the bound in Theorem 1 is dependent on $\hat{\epsilon}_{class}(T)$. Note however, that s is a combination of the context vector x_t and the classification function f_t. As $T \to \infty$, one would hope that f_t improves significantly and $\hat{\epsilon}_{class}(T)$ decays quickly. Thus, sublinear regret may still be achievable when f learns sufficiently quickly as a function of T. For instance, if f is optimizing a strongly convex loss function, online gradient descent achieves a regret guarantee of $O(\frac{\log T}{T})$ (Hazan et al., 2016, Theorem 3.3), potentially leading to a regret for MÊLÉE of $O(\sqrt{(\log T)/T})$.

The above statement is informal (it does not take into account the interaction between learning f and π). However, we can show a specific concrete example: we analyze MÊLÉE's test-time behavior when the underlying learning algorithm is BANDITRON. BANDITRON is a variant of the multiclass Perceptron that operates under bandit feedback. Details of this analysis (and proofs, which directly follow the original BANDITRON analysis) are given in Appendix B; here we state the main result. Let $\gamma_t = \Pr[r_t(\pi(f_t, x_t) = 1)|x_t] - \Pr[r_t(f_t(x_t)) = 1|x_t]$ be the edge of $\pi(f_t, .)$ over f, and let $\Gamma = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\frac{1}{1+K\gamma_t}$ be an overall measure of the edge. For instance if π simply returns f's prediction, then all $\gamma_t = 0$ and $\Gamma = 1$. We can then show the following:

B Stylized test-time analysis for Banditron: Details

The BANDITRONMÊLÉE algorithm is specified in Alg 2. The is exactly the same as the typical test time behavior, except it uses a BANDITRON-type strategy for learning the underlying classifier f in the place of POLOPT. POLICYELIMINATIONMETA takes as arguments: π (the learned exploration policy) and $\mu \in (0, 1/(2K))$ an added uniform exploration parameter. The BANDITRON learns a linear multi-class classifier parameterized by a weight matrix of size $K \times D$, where D is the input dimensionality. The BANDITRON assumes a pure multi-class setting in which the reward for one ("correct") action is 1 and the reward for all other actions is zero.

At each round t, a prediction $\hat{a}_t$ is made according to f_t (summarized by W^t). We then define an exploration distribution that "most of the time" acts according to $\pi(f_t, .)$, but smooths each action with μ probability. The chosen action a_t is sampled from this distribution and a binary reward is observed. The weights of the BANDITRON are updated according to the BANDITRON update rule using $\tilde{U}^t$.

Algorithm 2 BANDITRONMÊLÉE (g, μ)

initialize $W^1 = \mathbf{0} \in \mathbb{R}^{K \times D}$
for rounds $t = 1 \dots T$: **do**
  observe $x_t \in \mathbb{R}^D$
  compute $\hat{a}_t = f_t(x_t) = \text{argmax}_{k \in K} \left(W^t x_t\right)_k$
  define $Q^\mu(a) = \mu + (1 - K\mu)\mathbf{1}[a = \pi(W^t, x_t)]$
  sample $a_t \sim Q^\mu$
  observe reward $r_t(a_t) \in \{0, 1\}$
  define $\tilde{U}^t \in \mathbb{R}^{K \times D}$ as:
    $\tilde{U}^t_{a,\cdot} = x_t \left(\frac{\mathbf{1}[r_t(a_t)=1]\mathbf{1}[a_t=a]}{Q^\mu(a)} - \mathbf{1}[\hat{a}_t = a] \right)$
  update $W^{t+1} = W^t + \tilde{U}^t$
end for

The *only* difference between BANDITRONMÊLÉE and the original BANDITRON is the introduction of π in the sampling distribution. The original algorithm achieves the following mistake bound shown below, which depends on the notion of multi-class hinge-loss. In particular, the hinge-loss of W on $(x, \boldsymbol{r})$ is $\ell(W, (x, \boldsymbol{r})) = \max_{a \neq a^\star} \max\left\{0, 1 - (Wx)_{a^\star} + (Wx)_a\right\}$, where $a^\star$ is the a for which $r(a) = 1$. The overall hinge-loss L is the sum of ℓ over the sequence of examples.

Theorem 2 (Thm. 1 and Corr. 2 of Kakade et al. (2008)) *Assume that for the sequence of examples, $(x_1, \boldsymbol{r}_1), (x_2, \boldsymbol{r}_2), \dots, (x_T, \boldsymbol{r}_T)$, we have, for all t, $||x_t|| \leq 1$. Let $W^\star$ be any matrix, let L be the cumulative hinge-loss of $W^\star$, and let $D = 2\,||W^\star||_F^2$ be the complexity of $W^\star$. The number of mistakes M made by the* BANDITRON *satisfies*

$$\mathbb{E}M \leq L + K\mu T + 3\max\left\{\frac{D}{\mu}, \sqrt{DTK\mu}\right\} + \sqrt{DL/\mu} \tag{2}$$

where the expectation is taken with respect to the randomness of the algorithm. Furthermore, in a low noise setting (there exists $W^\star$ with fixed complexity d and loss $L \leq O(\sqrt{DKT})$), then by setting $\mu = \sqrt{D/(TK)}$, we obtain $\mathbb{E}M \leq O(\sqrt{KDT})$.

We can prove an analogous result for BANDITRONMÊLÉE. The key quantity that will control how much π improves the execution of BANDITRONMÊLÉE is how much π improves on f_t when f_t is wrong. In particular, let $\gamma_t = \Pr[r_t(\pi(f_t, x_t) = 1)|x_t] - \Pr[r_t(f_t(x_t)) = 1|x_t]$ be the edge of $\pi(f_t, .)$ over f, and let $\Gamma = \frac{1}{T}\sum_{t=1}^T \mathbb{E}\frac{1}{1+K\gamma_t}$ be an overall measure of the edge. (If π does nothing, then all $\gamma_t = 0$ and $\Gamma = 1$.) Given this quantity, we can prove the following Theorem 3.

Proof: [sketch] The proof is a small modification of the original proof of Theorem 2. The only change is that in the original proof, the following bound is used: $\mathbb{E}_t||\tilde{U}^t||^2/||x_t||^2 = 1 + 1/\mu \leq 2/\mu$. We use, instead: $\mathbb{E}_t||\tilde{U}^t||^2/||x_t||^2 \leq 1 + \mathbb{E}_t\frac{1}{\mu+\gamma_t} \leq \frac{2\mathbb{E}_t\frac{1}{1+\gamma_t}}{\mu}$. The rest of the proof goes through identically. □

Theorem 3 *Assume that for the sequence of examples, $(x_1, \boldsymbol{r}_1), (x_2, \boldsymbol{r}_2), \dots, (x_T, \boldsymbol{r}_T)$, we have, for all t, $||x_t|| \leq 1$. Let $W^\star$ be any matrix, let L be the cumulative hinge-loss of $W^\star$, let μ be a uniform exploration probability, and let $D = 2\,||W^\star||_F^2$ be the complexity of $W^\star$. Assume that $\mathbb{E}\gamma_t \geq 0$ for all t. Then the number of mistakes M made by* MÊLÉE *with* BANDITRON *as* POLOPT *satisfies:*

$$\mathbb{E}M \leq L + K\mu T + 3\max\left\{D\Gamma/\mu, \sqrt{DTK\Gamma\mu}\right\} + \sqrt{DL\Gamma/\mu} \tag{3}$$

where the expectation is taken with respect to the randomness of the algorithm.

Note that under the assumption $\mathbb{E}\gamma_t \geq 0$ for all t, we have $\Gamma \leq 1$. The analysis gives the same mistake bound for BANDITRON but with the factor of Γ, hence this result improves upon the BANDITRON analysis only when $\Gamma < 1$ (in the realizable setting, the number of mistakes is analogous to the regret). This result is highly stylized and the assumption that $\mathbb{E}\gamma_t \geq 0$ is overly strong. This assumption ensures that π never decreases the probability of a "correct" action. It does, however, help us understand the behavior of MÊLÉE, qualitatively: First, the quantity that matters in Theorem 3, $\mathbb{E}_t\gamma_t$ is (in the 0/1 loss case) exactly what MÊLÉE is optimizing: the expected improvement for

choosing an action against f_t's recommendation. Second, the benefit of using π within BANDITRON is a *local* benefit: because π is trained with expert rollouts, as discussed in Appendix A, the primary improvement in the analysis is to ensure that π does a better job predicting (in a single step) than f_t does. An obvious open question is whether it is possible to base the analysis on the *regret* of π (rather than its error) and whether it is possible to extend beyond the BANDITRON.

B.1 Training Details for the Exploration Policy

Exploration Features. In our experiments, the exploration policy is trained based on features Φ (Alg 1, line 10). These features are allowed to depend on the current classifier f_t, and on any part of the history *except* the inputs x_t in order to maintain task independence. We additionally ensure that its features are independent of the *dimensionality* of the inputs, so that π can generalize to datasets of arbitrary dimensions. The specific features we use are listed below; these are largely inspired by Konyushkova et al. (2017) but adapted and augmented to our setting.

The **features of** f_t that we use are: **a)** predicted probability $p(a_t|f_t, \boldsymbol{x}_t)$; **b)** entropy of the predicted probability distribution; **c)** a one-hot encoding for the predicted action $f_t(\boldsymbol{x}_t)$. The **features of** h_{t-1} that we use are: **a)** current time step t; **b)** normalized counts for all previous actions predicted so far; **c)** average observed rewards for each action; **d)** empirical variance of the observed rewards for each action in the history. In our experiments, we found that it is essential to calibrate the predicted probabilities of the classifier f_t. We use a very small held-out dataset, of size 30, to achieve this. We use Platt's scaling (Platt, 1999; Lin et al., 2007) method to calibrate the predicted probabilities. Platt's scaling works by fitting a logistic regression model to the classifier's predicted scores.

B.2 Details of Synthetic Datasets

Training Datasets. In our experiments, we follow Konyushkova et al. (2017) (and also Peters et al. (2014), in a different setting) and train the exploration policy π only on *synthetic data*. This is possible because the exploration policy π never makes use of x explicitly and instead only accesses it via f_t's behavior on it. We generate datasets with uniformly distributed class conditional distributions. The datasets are always two-dimensional. Details are in §B.2.

We generate datasets with uniformly distributed class conditional distributions. We generate 2D datasets by first sampling a random variable representing the Bayes classification error. The Bayes error is sampled uniformly from the interval 0.0 to 0.5. Next, we generate a balanced dataset where the data for each class lies within a unit rectangle and sampled uniformly. We overlap the sampling rectangular regions to generate a dataset with the desired Bayes error selected in the first step.

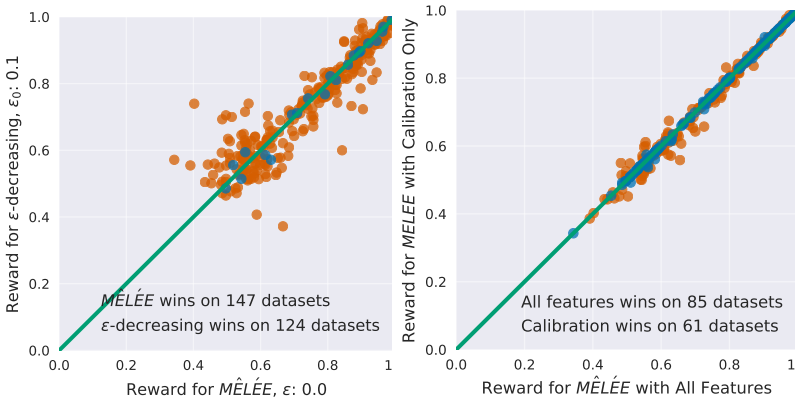
B.3 Implementation Details

Our implementation is based on scikit-learn (Pedregosa et al., 2011). We fix the training time exploration parameter μ to 0.1. We train the exploration policy π on 82 synthetic datasets each of size 3000 with uniform class conditional distributions, a total of $246k$ samples (§B.2). We train π using a linear classifier Breiman (2001) and set the hyper-parameters for the learning rate, and data scaling methods using three-fold cross-validation on the whole meta-training dataset. For the classifier class $\mathcal{F}$, we use a linear model trained with stochastic gradient descent. We standardize all features to zero mean and unit variance, or scale the features to lie between zero and one. To select between the two scaling methods, and tune the classifier's learning rate, we use three-fold cross-validation on a small fully supervised training set of size 30 samples. The same set is used to calibrate the predicted probabilities of f_t.

B.4 Baseline Exploration Algorithms

Our experiments aim to determine how MÊLÉE compares to other standard exploration strategies. In particular, we compare to:

- **ϵ-greedy:** With probability ϵ, explore uniformly at random; with probability $1 - \epsilon$ act greedily according to f_t (Sutton, 1996). Experimentally, we found $\epsilon = 0$ optimal on average, consistent with the results of Bietti et al. (2018).
- **ϵ-decreasing:** selects a random action with probabilities ϵ_i, where $\epsilon_i = \epsilon_0/t$, $\epsilon_0 \in]0, 1]$ and t is the index of the current round. In our experiments we set $\epsilon_0 = 0.1$. (Sutton & Barto, 1998)



(Left) MÊLÉE vs ϵ-decreasing; every point represents one dataset; the x-axis shows the reward of MÊLÉE, the y-axis shows ϵ-decreasing, and red dots represent statistically significant runs. **(Right)** MÊLÉE vs MÊLÉE using only the calibrated prediction probabilities (x-axis). MÊLÉE gets an additional leverage when using all the features.

Figure 2: Scatterplots comparing MÊLÉE to the best baseline and to a variant with fewer features.

Exponentiated Gradient ϵ-greedy: maintains a set of candidate values for ϵ-greedy exploration. At each iteration, it runs a sampling procedure to select a new ϵ from a finite set of candidates. The probabilities associated with the candidates are initialized uniformly and updated with the Exponentiated Gradient (EG) algorithm. Following Li et al. (2010b), we use the candidate set $\{\epsilon_i = 0.05{\times}i + 0.01, i = 1, \cdots, 10\}$ for ϵ.

LinUCB: Maintains confidence bounds for reward payoffs and selects actions with the highest confidence bound. It is impractical to run "as is" due to high-dimensional matrix inversions. We use diagonal approximation to the covariance when the dimensions exceeds 150. (Li et al., 2010a)

τ-first: Explore uniformly on the first τ fraction of the data; after that, act greedily.

Cover: Maintains a uniform distribution over a fixed number of policies. The policies are used to approximate a covering distribution over policies that are good for both exploration and exploitation (Agarwal et al., 2014).

Cover Non-Uniform: similar to Cover, but reduces the level of exploration of Cover to be more competitive with the Greedy method. Cover-Nu doesn't add extra exploration beyond the actions chose by the covering policies (Bietti et al., 2018).

In all cases, we select the best hyperparameters for each exploration algorithm following Bietti et al. (2018). These hyperparameters are: the choice of ϵ in ϵ-greedy, τ in τ-first, the number of bags, and the tolerance ψ for Cover and Cover-NU. We set $\epsilon = 0.0$, $\tau = 0.02$, bag size $= 16$, and $\psi = 0.1$.

B.5 Ablation Study

To understand more directly how MÊLÉE compares to ϵ-decreasing, in the left plot of Figure 2, we show a scatter plot of rewards achieved by MÊLÉE (x-axis) and ϵ-decreasing (y-axis) on each of the 300 datasets, with statistically significant differences highlighted in red and insignificant differences in blue. Points below the diagonal line correspond to better performance by MÊLÉE (147 datasets) and points above to ϵ-decreasing (124 datasets). The remaining 29 had no statistically significant difference.

We consider the effect that the additional features have on MÊLÉE's performance. In particular, we consider a version of MÊLÉE with all features (this is the version used in all other experiments) with an ablated version that only has access to the (calibrated) probabilities of each action from the underlying classifier f. The comparison is shown as a scatter plot in Figure 2 (right). Here, we can see that the full feature set *does* provide lift over just the calibrated probabilities, with a win-minus-loss improvement of 24 by adding additional features from which to learn to explore.