Meta-Learning Effective Exploration Strategies for Contextual Bandits

Amr Sharaf,[1] **Hal Daumé III**[1,2]
[1] University of Maryland
[2] Microsoft Research
sharaf@umd.edu, 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 uses imitation learning against these simulations to train an exploration policy that can be applied to true contextual bandit tasks at test time. We evaluate MÊLÉE on both a natural contextual bandit problem derived from a learning to rank dataset as well as hundreds of simulated contextual bandit problems derived from classification tasks. MÊLÉE outperforms seven strong baselines on most of these datasets by leveraging a rich feature representation for learning an exploration strategy.

1 Introduction

In a contextual bandit problem, an agent attempts to optimize its behavior over a sequence of rounds based on limited feedback (Kaelbling 1994; Auer 2003; Langford and 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 (§ 2). The feedback is *partial*: the agent only observes the reward for the action it selected, and for no other actions. This is strictly harder than supervised learning in which the agent observes the reward for *all* available actions. Contextual bandit problems arise in many real-world settings like learning to rank for information retrieval, online recommendations and personalized medicine. As in reinforcement learning, the agent must learn to balance *exploitation* (taking actions that, based on past experience, it believes will lead to high instantaneous reward) and *exploration* (trying new actions). However, contextual bandit learning is easier than reinforcement learning: the agent needs to only take one action, not a sequence of actions, and therefore does not face a credit assignment problem.

In this paper, we present a meta-learning approach to automatically learn a good exploration strategy from data. To achieve this, we use synthetic supervised learning data sets on which we can simulate contextual bandit tasks in an offline setting. Based on these simulations, our algorithm, MÊLÉE (MEta LEarner for Exploration), learns a good heuristic exploration strategy that generalize to future contextual bandit problems. MÊLÉE contrasts with more classical approaches to exploration (like ϵ-greedy or LinUCB), in which exploration strategies are constructed by expert algorithm designers. These approaches often achieve provably good exploration strategies in the worst case, but are potentially overly pessimistic and are sometimes computationally intractable.

MÊLÉE is an example of *meta-learning* in which we replace a hand-crafted learning algorithm with a learned learning algorithm. At training time (§ 3), MÊLÉE simulates many contextual bandit problems from fully labeled synthetic data. Using this data, in each round, MÊLÉE is able to counterfactually simulate what would happen under all possible action choices. We can then use this information to compute regret estimates for each action, which can be optimized using the AggreVaTe imitation learning algorithm (Ross and Bagnell 2014).

Our imitation learning strategy mirrors the meta-learning approach of Bachman, Sordoni, and Trischler (2017) in the active learning setting. We present a simplified, stylized analysis of the behavior of MÊLÉE to ensure that our cost function encourages good behavior (§4), and show that MÊLÉE enjoys the no-regret guarantees of the AggreVaTe imitation learning algorithm.

Empirically, we use MÊLÉE to train an exploration policy on only synthetic datasets and evaluate this policy on both a contextual bandit task based on a natural learning to rank dataset as well as three hundred simulated contextual bandit tasks (§5). We compare the trained policy to a number of alternative exploration algorithms, and show that MÊLÉE outperforms alternative exploration strategies in most settings.

2 Preliminaries: Contextual Bandits and Policy Optimization

Contextual bandits is a model of interaction in which an agent chooses actions (based on contexts) and receives immediate rewards for that action alone. For example, in a simplified news personalization setting, at each time step t, a user arrives and the system must choose a news article to

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

display to them. Each possible news article corresponds to an action a, and the user corresponds to a context x_t. After the system chooses an article a_t to display, it can observe, for instance, the amount of time that the user spends reading that article, which it can use as a reward $r_t(a_t)$.

Formally, we largely follow the setup and notation of Agarwal et al. (2014). Let $\mathcal{X}$ be an input space of contexts (users) and $[K] = \{1, \dots, K\}$ be a finite action space (articles). We consider the statistical setting in which there exists a fixed but unknown distribution $\mathcal{D}$ over pairs $(x, \boldsymbol{r}) \in \mathcal{X} \times [0,1]^K$, where $\boldsymbol{r}$ is a vector of rewards (for convenience, we assume all rewards are bounded in $[0,1]$). In this setting, the world operates iteratively over rounds $t = 1, 2, \dots$. Each round t:

1. The world draws $(x_t, \boldsymbol{r}_t) \sim \mathcal{D}$ and reveals context x_t.
2. The agent (randomly) chooses action $a_t \in [K]$ based on x_t, and observes reward $r_t(a_t)$.

The goal of an algorithm is to maximize the cumulative sum of rewards over time. Typically the primary quantity considered is the *average regret* of a sequence of actions $a_1, \dots, a_T$ to the behavior of the best possible function in a prespecified class $\mathcal{F}$:

$$\text{Reg}(a_1, \dots, a_T) = \max_{f \in \mathcal{F}} \frac{1}{T} \sum_{t=1}^{T} \Big[r_t(f(x_t)) - r_t(a_t) \Big] \quad (1)$$

A *no-regret* agent has a zero average regret in the limit of large T.

To produce a good agent for interacting with the world, we assume access to a function class $\mathcal{F}$ and to an *oracle policy optimizer* for that function class PolOpt. For example, $\mathcal{F}$ may be a set of single layer neural networks mapping user features $x \in \mathcal{X}$ to predicted rewards for actions $a \in [K]$. Formally, the observable record of interaction resulting from round t is the tuple $(x_t, a_t, r_t(a_t), p_t(a_t)) \in \mathcal{X} \times [K] \times [0,1] \times [0,1]$, where $p_t(a_t)$ is the probability that the agent chose action a_t, and the full history of interaction is $h_t = \langle (x_i, a_i, r_i(a_i), p_i(a_i)) \rangle_{i=1}^{t}$. The oracle policy optimizer, PolOpt, takes as input a *history* of user interactions and outputs an $f \in \mathcal{F}$ with low expected regret.

An example of the oracle policy optimizer PolOpt is to combine inverse propensity scaling (IPS) with a regression algorithm (Horvitz and Thompson 1952). Here, given a history h, each tuple (x, a, r, p) in that history is mapped to a multiple-output regression example. The input for this regression example is the same x; the output is a vector of K costs, all of which are zero except the a^{th} component, which takes value r/p. This mapping is done for all tuples in the history, and a supervised learning algorithm on the function class $\mathcal{F}$ is used to produce a low-regret regression function f. This is the function returned by the oracle policy optimizer PolOpt. IPS has the property of being unbiased, however, it often suffers from large variance. The direct method (DM) (Dudik et al. 2011) is another kind of an oracle policy optimizer PolOpt that has lower variance than IPS. The direct method estimates the reward function directly from the history h without importance sampling, and uses this estimate to learn a low-regret function f. In our experiments, we use the direct method, largely for its low variance and simplicity. However, MêLée is agnostic to the type of the estimator used by the oracle policy optimizer PolOpt.

3 Approach: Learning an 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 a strategy like ϵ-greedy by taking more (any!) information into account; for instance, basing the probability of exploration on f's uncertainty. In this section, we describe MêLée, first by showing how it operates in a Markov Decision Process (§3), and then by showing how to train it using synthetic simulated contextual bandit problems based on imitation learning (§3).

Markov Decision Process Formulation

We model the exploration / exploitation task as a Markov Decision Process (MDP). Given a context vector x and a function f output by PolOpt on rounds $1 \dots t-1$, the agent learns an exploration policy π to decide whether to *exploit* by acting according to $f(x)$, or *explore* by choosing a different action $a \neq f(x)$. We model this as an MDP, represented as a tuple $\langle A, S, s_0, T, R \rangle$, where $A = \{a\}$ is the set of all actions, $S = \{s\}$ is the space of all possible states, s_0 is a starting state (or initial distribution), $R(s, a)$ is the reward function, and $T(s'|s, a)$ is the transition function. We describe these below.

States A state s in our MDP represents all past information seen in the world (there are a very large number of states). At time t, the state s_t includes: all past experiences $(x_i, a_i, r_i(a_i))_{i=1}^{t-1}$, the current context x_t, and the current function f_t computed by running PolOpt on the past experiences.

For the purposes of learning, this state space is far too large to be practical, so we model each state using a set of exploration features. The exploration policy π is trained based on exploration features Φ (Alg 1, line 12). 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, Sznitman, and Fua (2017) but adapted to our setting.

Importantly, we wish to train π using one set of tasks (for which we have fully supervised data on which to run simulations) and apply it to wholly different tasks (for which we only have bandit feedback). To achieve this, we allow π to depend representationally on f_t in arbitrary ways: for instance, it might use features that capture f_t's uncertainty on the current example. We additionally allow π to depend in a *task-independent* manner on the history (for instance, which actions have not yet been tried): it can use features of the actions, rewards and probabilities in the history but *not* depend directly on the contexts x. This is to ensure that π only learns to explore and not also to solve the underlying task-dependent classification problem. Because π needs to learn to be task independent, we found that if f_t's predictions were uncalibrated, it was very difficult for π to generalize well to unseen tasks. Therefore, we additionally allow π to depend on a *very small* amount of fully labeled data from the task at hand, which we use to allow π to calibrate f_t's predictions. In our experiments we use only 30 fully labeled examples, but alternative approaches to calibrating f_t that do not require this data would be preferable. The features of f_t that we use are: **a)** predicted probability $p(a_t|f_t, \boldsymbol{x}_t)$, we use a softmax over the predicted rewards from f_t to convert them to probabilities; **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.

We use Platt's scaling (Platt 1999; Lin, Lin, and Weng 2007) method to calibrate the predicted probabilities. Platt's scaling works by fitting a logistic regression model to the classifier's predicted scores.

Initial State The intial state distribution is formed by drawing a new contextual bandit task at random, setting the history to the empty set, and initializing the first context x_1 as the first example in that contextual bandit task.

Actions At each state, our learned exploration policy π must take an input state s_t (described above) and make a decision. Its action space A is the same action space as that of the contextual bandit problem it is trying to solve. If π chooses to take the same action as f, then we interpret this as an "exploitation" step, and if it takes another action, we interpret this as an "exploration" step.

Transitions Each episode starts off with a new contextual bandit task and an empty history $h_0 = \{\}$. The subsequent steps in the episode involve observing context vectors $x_1, \cdots, x_T$ from the new contextual bandit task. A single transition in the episode consists of the exploration policy π being given the state s containing information about the current context vector x_t and the history h_{t-1}, using which, the exploration policy π chooses the next action a. The transition function $T(s'|s, a)$ incorporates the action a chosen by the exploration policy in state s along with the features representing the current state s, and produces the next state s' that represents a new feature vector x_{t+1}. The episode terminates whenever all the context vectors in the contextual bandit task have been exhausted. During the test phase, each contextual bandit task is handled only once in a single episode.

More formally, let π be the exploration policy we are learning, which takes *two inputs*: a function $f \in \mathcal{F}$ and a context x, and outputs an action. In our example, f will be the output of the policy optimizer on all historic data, and x will be the current user. 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 $f_t \leftarrow \textsc{PolOpt}(h_t)$ 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 $r_t(a_t)$.
5. The agent appends $(x_t, a_t, r_t(a_t), p_t)$ to the history h_t, 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 (§ 5), we find that setting $\mu = 0$ is optimal in aggregate, but non-zero μ is necessary for our theory (§4).

Rewards The reward function is chosen so that reward maximization by the learned policy is equivalent to low regret in the contextual bandit problem. Formally, at state s_t, let $r_t(\cdot)$ be the reward function for the contextual bandit task at that state. The reward function is: $R(s, a) = r_t(a)$.

Training MÊLÉE by Imitation Learning

The meta-learning challenge is: how do we learn a good exploration policy π? We assume we have 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 as this training data; one could alternatively use any fully labeled classification dataset as in (Beygelzimer and Langford 2009). Under this assumption about the data, and with our model of behavior as an MDP, a natural class of learning algorithms to consider for learning are imitation learning algorithms (Daumé, Langford, and Marcu 2009; Ross, Gordon, and Bagnell 2011; Ross and Bagnell 2014; Chang et al. 2015). 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.

Informally, at training time, MÊLÉE will treat one of these synthetic datasets as if it were a contextual bandit dataset. At each time step t, it will compute f_t by running $\textsc{PolOpt}$ on the historical data, and then consider: for *each* action, what would the long time reward look like if I were to take this action. Because the training data for MÊLÉE is fully labeled, this can be evaluated for each possible action, and a policy π can be learned to maximize these rewards.

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

```
1: initialize meta-dataset $D = \{\}$
2: for episode $n = 1, 2, \ldots, N$ do
3:     choose $S$ at random from $\{S_m\}$, and set history $h_0 = \{\}$
4:     partition and permute $S$ randomly into train $Tr$ and validation $Val$ where $|Val| = N_{Val}$
5:     for round $t = 1, 2, \ldots, |Tr|$ do
6:         let $(x_t, \boldsymbol{r}_t) = Tr_t$, $s_t = (x_i, a_i, r_i(a_i))_{i=1}^{t-1}, x_t, f_t$
7:         for each action $a = 1, \ldots, K$ do
8:             $f_{t,a} = \textsc{PolOpt}(\mathcal{F}, h_{t-1} \oplus (x_t, a, r_t(a), 1 - \frac{K-1}{K}\mu))$ on augmented history
9:             roll-out: estimate $Q^\star(s_t, a)$, the cost-to-go of $a$, using $r_t(a)$ and a roll-out policy $\pi^{\text{out}}$ on $f_{t,a}$
10:        end for
11:        compute $f_t = \textsc{PolOpt}(\mathcal{F}, h_{t-1})$
12:        $D \leftarrow D \oplus (\Phi(s_t), \langle Q^\star(s_t, 1), \ldots, Q^\star(s_t, K)\rangle)$
13:        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
14:        append history $h_t \leftarrow h_{t-1} \oplus (x_t, a_t, r_t(a_t), p_t)$
15:    end for
16:    update $\pi_n = \textsc{Learn}(D)$
17: end for
18: return best policy in $\{\pi_n\}_{n=1}^{N}$
```

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. Because we can train on fully supervised training sets, we can easily define an optimal reference policy $\pi^\star$, which "cheats" at training time by looking at the true labels: in particular, $\pi^\star$ can always pick the correct action (i.e., the action that maximizes future rewards) at any given state. The learning problem is then to estimate π to have as similar behavior to $\pi^\star$ as possible, but without access to those labels.

Suppose we wish to learn an exploration policy π for a contextual bandit problem with K actions. We assume access to M supervised learning datasets $S_1, \ldots, S_M$, where each $S_m = \{(x_1, \boldsymbol{r}_1), \ldots, (x_{N_m}, \boldsymbol{r}_{N_m})\}$ of size N_m, where each x_n is from a (possibly different) input space $\mathcal{X}_m$ and the reward vectors are all in $[0, 1]^K$. In particular, multi-class classification problems are modeled by setting the reward for the correct label to be one and the reward for all other labels to be zero.

The imitation learning algorithm we use is AggreVaTe (Ross and Bagnell 2014) (closely related to DAgger (Ross, Gordon, and Bagnell 2011)), and is instantiated for the contextual bandits meta-learning problem in 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. On the first iteration AggreVaTe collects data by observing the expert perform the task, and in each trajectory, at time t, explores an action a in state s, and observes the cost-to-go $Q_t^\star(s, a)$ of the expert after performing this action, defined as:

$$Q_t^\star(s, a) = r_t(a) + E_{s\sim T(.|s,a), a\sim\pi^\star(.|s)}[Q_{t+1}^\star(s, a)]. \quad (2)$$

where the expectation is taken over the randomness of the policy $\pi^\star$ and the MDP.

Each such step generates a cost-weighted training example $(s, t, a, Q^\star)$ and AggreVaTe trains a policy π_1 to minimize the expected cost-to-go on this dataset. At each following iteration n, AggreVaTe collects data through interaction with the learner as follows: for each trajectory, begin by using the current learner's policy π_n to perform the task, interrupt at time t, explore a roll-in action a in the current state s, after which control is provided back to the expert to continue up to time-horizon T. This results in new examples of the cost-to-go (roll-out value) of the expert $(s, t, a, Q^\star)$, under the distribution of states visited by the current policy π_n. This new data is aggregated with all previous data to train the next policy π_{n+1}; more generally, this data can be used by a no-regret online learner to update the policy and obtain π_{n+1}. This is iterated for some number of iterations N and the best policy found is returned.

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 episodes, 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 4). 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 8) on which it trains a new policy and evaluates it's **roll-out value** (line 9). This yields, for each t, a new training example for π, which is added to π's training set (line 12); the features for this example are features of the classifier based on true history (line 11) (and possibly statistics of the history itself), with a label that gives, for each action, the corresponding cost-to-go of that action (the $Q^\star$s computed in line 9). MÊLÉE then must commit to a **roll-in action** to *actually* take; it chooses this according to a roll-in policy (line 13). MÊLÉE has no explicit "exploitation policy", exploitation happens when π chooses the same action as f_t, while exploration happens when it chooses a different action. In learning to explore, MÊLÉE simultaneously learns when to exploit.

Roll-in actions. The distribution over states visited by MÊLÉE depends on the actions taken, and in general it is good to have that distribution match what is seen at test time. This distribution is determined by a *roll-in* policy (line 13), controlled in MÊLÉE by exploration parameter $\mu \in [0, 1]$. As $\mu \to 1$, the roll-in policy approaches a uniform random policy; as $\mu \to 0$, the roll-in policy becomes deterministic. When the roll-in policy does not explore, it acts according to $\pi(f_t, .)$.

Roll-out values. The ideal value to assign to an action (from the perspective of the imitation learning procedure) is that total reward (or advantage) that would be achieved in the long run if we took this action and then behaved according to our final learned policy. Unfortunately, during training, we do not yet know the final learned policy. Thus, a surrogate roll-out policy π^{out} is used instead. A convenient, and often computationally efficient alternative, is to evaluate the value assuming all future actions were taken by the expert (Langford and Zadrozny 2005; Daumé, Langford, and Marcu 2009; Ross and Bagnell 2014). In our setting, at any time step t, the expert has access to the fully supervised reward vector $\boldsymbol{r}_t$ for the context $\boldsymbol{x}_t$. When estimating the roll-out value for an action a, the expert will return the true reward value for this action $r_t(a)$ and we use this as our estimate for the roll-out value.

4 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 and Bagnell 2014); as such, it inherits AGGREVATE's regret guarantees.

Theorem 1 *After N episodes, if* LEARN *(line 16) is no-regret algorithm, then as $N \to \infty$, with probability 1, it holds that $J(\bar{\pi}) \geq J(\pi^\star) - 2T\sqrt{K\hat{\epsilon}_{class}(T)}$, 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 $Q^\star$, where*

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

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 $t \sim U(T)$, $s \sim d^t_{\pi_i}$, $U(T)$ the uniform distribution over $\{1, \dots, T\}$, 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

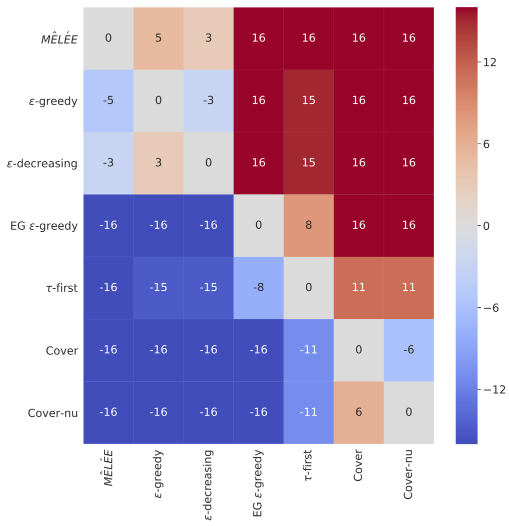

Figure 1: Win/Loss counts for all pairs of algorithms over 16 random shuffles for the MSLR-10K dataset.

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 the appendix.

5 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 (§ 3). Once π is learned and fixed, we follow the test-time behavior described in § 3 to evaluate π on a set of contextual bandit problems. We evaluate MÊLÉE on a natural learning to rank task (§5). To ensure that the performance of MÊLÉE generalizes beyond this single learning to rank task, we additionally perform thorough evaluation on 300 "simulated" contextual bandit problems, derived from standard classification task.

In all cases, the underlying classifier f is a linear model trained with an optimizer that runs stochastic gradient descent. 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?

Training Datasets

In our experiments, we follow Konyushkova, Sznitman, and Fua (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 dis-

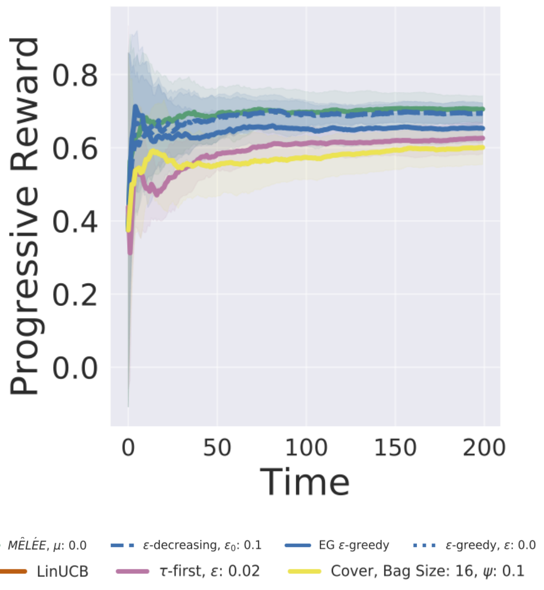

Figure 2: Learning curve on the MSLR-10K dataset: x-axis shows the number of queries observed, and y-axis shows the progressive reward.

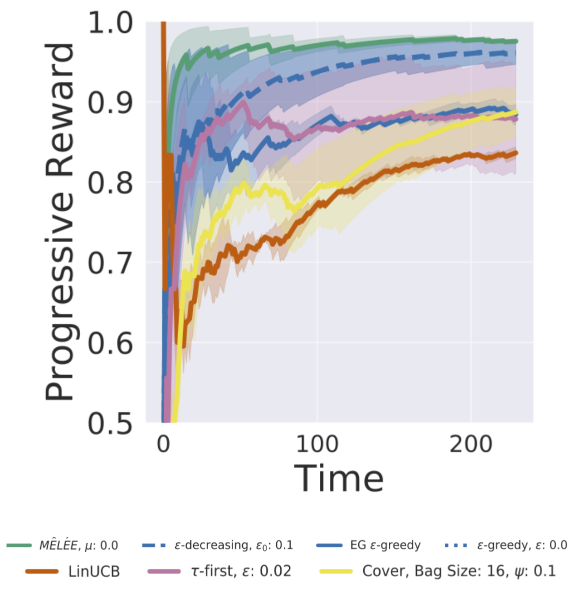

Figure 3: Behavior of MÊLÉE in comparison to baseline and state-of-the-art exploration algorithms. A representative learning curve on dataset #1144.

tributed class conditional distributions. The datasets are always two-dimensional.

Evaluation Methodology

For evaluation, we use progressive validation (Blum, Kalai, and Langford 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. Progressive validation is particularly suitable for measuring the effectiveness of the exploration algorithm, since the decision on whether to exploit or explore at earlier time steps will affect the performance on the observed examples in the future.

Because our evaluation is over 300 datasets, we report aggregate results in terms of **Win/Loss Statistics:** We compare two exploration methods 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.

Experimental Results

Learning to Rank We evaluate MÊLÉE on a natural learning to rank dataset. The dataset we consider is the Microsoft Learning to Rank dataset, variant MSLR-10K from (Qin and Liu 2013). The dataset consists of feature vectors extracted from query-url pairs along with relevance judgment labels. The relevance judgments are obtained from a retired labeling set of a commercial web search engine (Microsoft Bing), which take 5 values from 0 (irrelevant) to 4 (perfectly relevant) . In our experiments, we limit the number of labels to the two extremes: 0 and 4, and we drop the queries not labelled as any of the two extremes. A query-url pair is represented by a 136-dimensional feature vector. The dataset is highly imbalanced as the number of irrelevant queries is much larger than the number of relevant ones. To address this, we sample the number of irrelevant queries to match that of the relevant ones. To avoid correlations between the observed query-url pairs, we group the queries by the query ID, and sample a single query from each group. We convert relevance scores to losses with 0 indicating a perfectly relevant document, and 1 an irrelevant one.

Figure 2 shows the evaluation results on a subset of the MSLR-10K dataset. Since the performance is closely matched between the different exploration algorithms, we repeat the experiment 16 times with randomly shuffled permutations of the MSLR-10K dataset. Figure 2 shows the learning curve of the trained policy π as well as the baselines. Here, we see that MÊLÉE quickly achieves high reward, after about 100 examples the two strongest baselines catch up. By 200 examples all approaches have asymptoted. We exclude LinUCB from these runs because the required matrix inversions made it too computationally expensive.[1] Figure 1 shows statistically-significant win/loss differences for each of the algorithms, across these 16 shuffles. 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

[1]In a single run of LinUCB we observed that its performance is on par with ϵ-greedy.

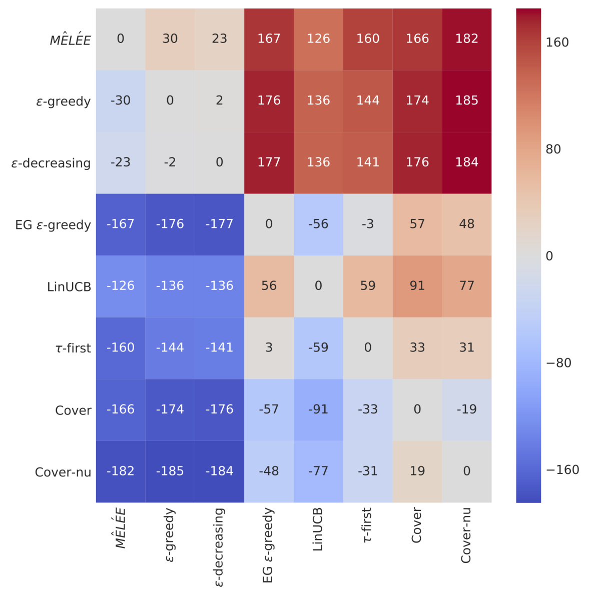

Figure 4: Behavior of MÊLÉE in comparison to baseline and state-of-the-art exploration algorithms. Win statistics: each (row, column) entry shows the number of times the row algorithm won against the column, minus the number of losses. MÊLÉE outperforms the nearest competition (ϵ-decreasing) by 23.

competition (ϵ-decreasing) by 3 points.

Simulated Contextual Bandit Tasks We perform an exhaustive evaluation on simulated contextual bandit tasks to ensure that the performance of MÊLÉE generalizes beyond learning to rank. Following Bietti, Agarwal, and Langford (2018), we use a collection of 300 binary classification datasets from openml.org for evaluation. These datasets cover a variety of different domains including text & image processing, medical, and sensory data. We convert classification datasets into cost-sensitive classification problems by using a $0/1$ encoding. Given these fully supervised cost-sensitive multi-class datasets, we simulate the contextual bandit setting by only revealing the reward for the selected actions.

In Figure 3, 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 4, 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.

To understand more directly how MÊLÉE compares to ϵ-

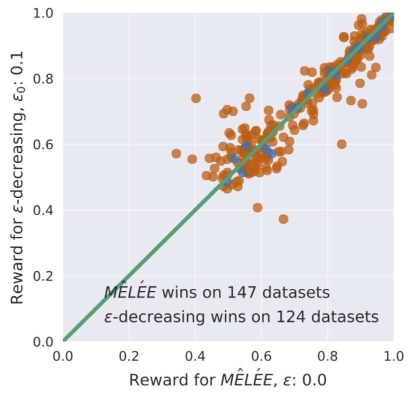

Figure 5: MÊLÉE vs ϵ-decreasing; every point represents one dataset; the x-axis shows the reward of MÊLÉE, the y-axis shows ϵ-decreasing, and red dots represent statistically significant runs.

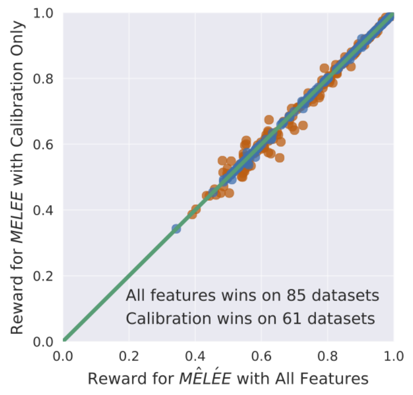

Figure 6: MÊLÉE vs MÊLÉE using only the calibrated prediction probabilities (x-axis). MÊLÉE gets an additional leverage when using all the features.

decreasing, in Figure 5, 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.

Finally, 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 6. 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.

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. 2014. Taming the monster: A fast and simple algorithm for contextual bandits. In *In Proceedings of the 31st International Conference on Machine Learning (ICML-14*, 1638–1646.

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

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

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

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

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

Chang, K.; Krishnamurthy, A.; Agarwal, A.; Daumé, III, H.; and Langford, J. 2015. Learning to Search Better Than Your Teacher. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML, 2058–2066. JMLR.org.

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

Dudik, M.; Hsu, D.; Kale, S.; Karampatziakis, N.; Langford, J.; Reyzin, L.; and Zhang, T. 2011. Efficient optimal learning for contextual bandits. *arXiv preprint arXiv:1106.2369* .

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

Horvitz, D. G.; and Thompson, D. J. 1952. A Generalization of Sampling Without Replacement from a Finite Universe. *Journal of the American Statistical Association* 47(260): 663–685. doi:10.1080/01621459.1952.10483446.

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

Konyushkova, K.; Sznitman, R.; and Fua, P. 2017. Learning Active Learning from Data. In *Advances in Neural Information Processing Systems*.

Langford, J.; and Zadrozny, B. 2005. Relating reinforcement learning performance to classification performance. In *Proceedings of the 22nd international conference on Machine learning*, 473–480. ACM.

Langford, J.; and Zhang, T. 2008. The Epoch-Greedy Algorithm for Multi-armed Bandits with Side Information. In *Advances in Neural Information Processing Systems 20*, 817–824. Curran Associates, Inc.

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

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

Platt, J. C. 1999. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, 61–74. MIT Press.

Qin, T.; and Liu, T. 2013. Introducing LETOR 4.0 Datasets. *CoRR* abs/1306.2597. URL http://arxiv.org/abs/1306.2597.

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

Ross, S.; Gordon, G.; and Bagnell, J. A. 2011. 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*, 627–635. Fort Lauderdale, FL, USA: PMLR.