Constrained Policy Optimization for Controlled Contextual Bandit Exploration

Mohammad Kachuee, Sungjin Lee

Amazon Alexa AI, Bellevue, WA 98004, USA

Abstract

Contextual bandits are widely used across the industry in many applications such as search engines, dialogue systems, recommendation systems, etc. In such applications, it is often necessary to update the policy regularly as the data distribution changes and new features are being on-boarded frequently. As any new policy deployment directly impacts the user experience, safety in model updates is an important consideration in real-world bandit learning. In this study, we introduce a scalable framework for policy update safety via user-defined constraints, supporting fine-grained exploration targets for individual domains. For example, in a digital voice assistant, we may want to ensure fewer policy deviations in business-critical domains such as shopping, while allocating more exploration budget to domains such as music. Furthermore, we present a novel meta-gradient learning method that is scalable and practical to address this problem. The proposed method adjusts constraint violation penalty terms adaptively through a meta objective that encourages balanced constraint satisfaction across domains. We conduct extensive experiments using data from a real-world conversational AI system on a set of realistic constraint benchmarks. Based on the experimental results, we demonstrate that the proposed approach is capable of achieving the best balance between the policy value and constraint satisfaction rate.

Keywords

contextual bandit, constrained optimization, safe exploration

1. Introduction

Contextual bandits are important machine learning problems used in many real-world applications such as search engines, advertisement systems, conversational systems, etc. In a contextual bandit problem, the agent is tasked to select the action that achieves the highest reward given a context. In this setting, the environment is assumed to have no memory, and the reward is a stochastic function of the current context, independent of past agent-environment interactions [1, 2, 3].

For real-world bandits, ensuring safe policy updates is a crucial consideration. Although there have been a number of prior studies under safe bandit updates in an online learning setting, online bandit learning is not suitable for business-critical production systems where an updated policy must go through extensive testing to ensure reliable performance on critical use cases before it gets deployed. This calls out for an approach for safe policy updates in the offline bandit learning setting.

In a production system, it is crucial to not only estimate but also control the changes of behavior a new policy introduces when compared to the current production policy. In the literature, this problem has been studied under safe bandit updates [4, 5, 6] and budgeted bandit learning [7, 8], usually targeting exploration budgets or encouraging a behavior resembling a baseline policy.

In the context of off-policy bandit updates, we define exploration as any change in the model behavior resulting from replacing a current production policy with a new updated policy. This definition is broad and encloses stochastic exploration actions as well as any behavior change when comparing the two consecutive policies.

Furthermore, we consider the scenario in which samples are naturally classified into a set of domains, each representing a unique data segment. For example, when dealing with product reviews, the product category (e.g., books, computers, etc.) would be the domain for each review. In many real-world scenarios, it is desirable to have fine-grained control on the rate of exploration for each domain. For example, in a conversational system, we may want to explore business-critical domains such as shopping more conservatively, while aggressively exploring a domain such as entertainment. Note that solely relying on a method such as Thompson sampling [9] or epsilon-greedy [10] would neither be effective to meet the requirements for individual domains nor optimal to balance between the desired fine-grained exploration and achieving the highest possible reward.

While previous studies considered different aspects of constraining a bandit model, to the best of our knowledge the problem of controlling off-policy bandit behavior changes across subsequent model updates with a fine-grained control on budgets for different data segments (domains) remains unaddressed. This study is the first to tackle the aforementioned issues by providing a scalable

The IJCAI-ECAI-22 Workshop on Artificial Intelligence Safety (AISafety 2022), July 24–25, 2022, Vienna, Austria

kachum@amazon.com (M. Kachuee); sungjinl@amazon.com (S. Lee)

0000-0002-0099-3466 (M. Kachuee)

and practical approach. The main contributions of this paper are as follows:

- Introducing a formulation for controlled exploration in the context of off-policy contextual bandit learning considering fine-grained control over domains based on user-defined constraints.
- Presenting a solution based on the primal-dual minimax constraint optimization method that is effective but requires adjusting a few hyperparameters.
- Proposing a novel meta gradient algorithm to balance constraint satisfaction and reward maximization that works out-of-the-box and outperforms other methods with no need for hyperparameter adjustment.
- Conducting extensive experiments on the skill routing problem in a real-world conversational AI agent using a set of realistic constraint benchmarks.

2. Related Work

2.1. Constrained Bandit Learning

The majority of studies on controlled bandit learning consider the case of simple multi-armed stochastic bandits (i.e., without context) with practical applications in experiment design [8] and automated machine learning [7]. Hoffman et al. [7] suggested a Bayesian approach to two-phase exploration-exploitation bandit learning in which there is a pre-specified budget for exploration arm evaluations. Another aspect is to ensure safe exploration actions, which is especially useful for sensitive applications in industrial machine learning or healthcare. Amani et al. [6] introduced a solution in which an initial set of exploration actions is defined, then the exploration set is gradually expanded to ensure minimal unexpected behavior.

For contextual bandits, safety has been an active research topic. Safety can be defined in the action space or in terms of model updates. For example, Daulton et al. [5] solves a two-metric setting in which one of the metrics, reward, is being maximized while enforcing a limit for regression on an auxiliary metric compared to a baseline status quo model. Balakrishnan et al. [11] attempts to learn behavioral constraints by balancing between replication of the current baseline policy and making new actions that show promising rewards. In [4] authors define safety in terms of user experience metrics and suggest deciding on deploying a new model based on conservative confidence bounds on the off-policy estimates of such metrics.

2.2. Constrained Optimization

The general problem of constrained optimization has been studied extensively in the literature. The type of constraints and the optimization problem are important aspects in the design of such algorithms. In the context of constrained optimization of differentiable function approximators (e.g., neural networks) the constraints can be defined on the parameters [12], outputs [13], or based on predefined functions of the model [14, 15].

Most relevant to this paper, penalty methods translate constraints into penalty terms used to encourage constraint satisfaction in the optimization process. The quadratic penalty method adds the weighted square of violation metrics and adjusts the weight either by heuristics or solving a sequence of problems with a monotonically increasing penalty weight. The augmented Lagrangian method leverages Lagrange multipliers to mitigate the issue of ill-conditioning inherent in the quadratic penalty method [16, 17]. More applicable to neural network models, Nandwani et al. [14] suggested a scalable approach to defining constraints using hinge functions, then solving the dual minimax problem. Theoretically, given an infinite number of training iterations and a decaying update rate for the max variables, it is known that the minimax objective is guaranteed to converge to a local min-max point [18]. Nonetheless, such a guarantee does not apply to real-world settings in which a non-convex problem is being solved in finite time using limited compute resources.

2.3. Meta Learning

In the literature, meta-learning is defined as learning to learn in which the model training itself is considered an inner optimization process guided by an outer meta objective. It is a broad topic spanning multiple areas of research including reward function discovery in reinforcement learning [19, 20], life-long continual learning [21], few-shot learning [22], and transfer learning [23] to name a few.

Most relevant to this work is the meta-gradient idea first suggested by Sutton [24]. In meta-gradient learning, we use online cross-validation on a meta objective to compute derivatives with respect to meta parameters through a differentiable inner optimization loop. Xu et al. [25] demonstrated the effectiveness of this meta-gradient idea in deep reinforcement learning to adapt the value function as the agent is interacting with the environment. However, in practice, the compute/memory requirement for computing high-order gradients is a major challenge for meta-reinforcement learning leading to low-order gradient approximations [26, 25]. Also, while meta-gradient learning enables adaptive adjustment of hyperparameters, it still requires a carefully designed meta objective

that provides the best learning signal.

3. Constrained Bandit Exploration

3.1. Problem Formulation

We consider the general framework of off-policy contextual bandits in which a policy Π is used to select an action $a \in A$ given the observed context vector ($\mathbf{x}$) to maximize the scalar reward (r) received from the environment. Here, we assume stochastic policies of the form $\Pi_\theta(a|\mathbf{x})$ in which a model parameterized by θ (e.g., a neural network) is used to assign action selection probabilities to each action given the context. Furthermore, we assume that each sample belongs to a domain denoted by $k \in 1 \dots M$ that is provided as a feature in $\mathbf{x}$.

In the off-policy setting, the policy is refreshed after collecting a dataset of samples from the current policy. We adopt a definition of exploration which considers any change in the agent behavior compared to the current policy as an exploration action. Alternatively, we can consider replication with respect to the current policy as the rate at which the new policy makes similar decisions to the current policy when both evaluated and sampled stochastically. We define replication for Π_θ with respect to Π_0 based on the L1-distance of their action propensities given a context $\mathbf{x}$:

$$\mathcal{R}_\theta(\mathbf{x}) = 1 - \frac{|\Pi_\theta(\mathbf{x}) - \Pi_0(\mathbf{x})|_1}{2} \ . \tag{1}$$

In a production system, it is desirable to precisely control the rate at which the new policy replicates the current policy for each domain. This ensures safe model updates for critical domains while enabling exploration for others that may benefit from an extra exploration budget. Accordingly, we define constraints to encourage the desired behavior for samples of each domain, while learning an off-policy bandit:

$$\begin{aligned} &\arg\min_\theta \ \ \mathbb{E}_{\mathbf{x},a,r,k\sim\mathbb{D}} \ \ L_{\Pi_\theta} \ \ , \\ &s.t. \ \ \ c_k^{min} \le \mathcal{R}_\theta(\mathbf{x}) \le c_k^{max} \end{aligned} \tag{2}$$

where context, action, reward, and domain $(\mathbf{x}, a, r, k)$ are sampled from a dataset collected from the current policy. In (2), we use c_k^{min} and c_k^{max} to indicate user-defined replication constraints for domain k.

L_{Π_θ} can be any differentiable off-policy bandit learning objective, for simplicity of discussion, we consider the vanilla inverse propensity scoring (IPS) objective:

$$L_{\Pi_\theta}(\mathbf{x}, a, r) = -r\frac{\Pi_\theta(a|\mathbf{x})}{\Pi_0(a|\mathbf{x})} \ \ , \tag{3}$$

where Π_0 is the current policy and r is the observed reward for taking action a collected in the dataset.

A common approach to optimize constrained problems such as (2) is to use the penalty method, translating constraints into penalty terms that encourage constraint satisfaction:

$$\begin{aligned} &\arg\min_\theta \mathbb{E}_{\mathbf{x},a,r,k\sim\mathbb{D}}[L_{\Pi_\theta}(\mathbf{x}, a, r) + \\ &e^{\mathbf{u}_k} \max(0, c_k^{min} - \mathcal{R}_\theta(\mathbf{x})) + e^{\mathbf{v}_k} \max(0, \mathcal{R}_\theta(\mathbf{x}) - c_k^{max})]. \end{aligned} \tag{4}$$

Here, penalty terms are always non-negative and increase if the new policy assigns action probabilities that deviate from the current policy outside the desired boundary. $\mathbf{u} \in R^M$ and $\mathbf{v} \in R^M$ are variables that adjust the weight of each constraint violation term. The exponentiation improves the sensitivity to these parameters and ensures having non-negative penalty terms. For (4) to actually solve the original constrained problem of (2), proper values for $\mathbf{u}$ and $\mathbf{v}$ need to be used that enable the best balance between constraint satisfaction and the policy value. In the constrained optimization literature, various methods have been suggested to solve this form of problem (see Section 2.2). In this paper, to solve this problem, we use the primal-dual minimax method suggested by Nandwani et al. [14] (Section 3.2) as well as a novel meta-learning method (Section 3.3).

3.2. Minimax Primal-Dual Method

Nandwani et al. [14] suggested a formulation of the augmented Lagrangian method that supports inequality constraints. They solve the dual problem which is optimizing the dual maximin problem to improve the scalability:

$$\begin{aligned} &\min_\theta \max_{\mathbf{u},\mathbf{v}} \mathbb{E}_{\mathbf{x},a,r,k\sim\mathbb{D}}[L_{\Pi_\theta}(\mathbf{x}, a, r) + \\ &e^{\mathbf{u}_k} \max(0, c_k^{min} - \mathcal{R}_\theta(\mathbf{x})) + e^{\mathbf{v}_k} \max(0, \mathcal{R}_\theta(\mathbf{x}) - c_k^{max})]. \end{aligned} \tag{5}$$

Algorithm 1 shows an outline of the policy training using the minimax method. This method has four hyperparameters controlling the max player optimization via adjusting the update frequency, learning rate, and decay factors.

Intuitively, the min player is trying to update the policy parameters while the max player is increasingly penalizing it for any constraint violation. A stable point of this algorithm would be to gradually reduce the max player update rate as the min player is getting better at satisfying the constraints, eventually satisfying all constraints resulting in a zero loss for the max player due to the zero hinge penalty terms.

3.3. Meta Gradient Method

Theoretically, the primal-dual minimax method is capable of achieving Pareto optimal solutions [18, 14]. However, in practice, it is infeasible to train for an infinite

Algorithm 1: Minimax constrained bandit optimization algorithm

input : $\mathbb{D}$ (dataset), η (max learning rate), γ (max learning rate decay), τ (max update frequency), ξ (max update frequency decay)

$\mathbf{u}, \mathbf{v}, t \leftarrow 0$
Initialize(Π_θ)
for $\mathbf{x}, a, r, k \sim D$ **do**
  /* use loss function of (5) */
  $L \leftarrow Loss(\mathbf{x}, a, r, k, \theta, \mathbf{u}, \mathbf{v})$
  if *$t\%\tau$ is 0* **then**
    /* gradient ascent, max player */
    $\mathbf{u} \leftarrow \mathbf{u} + \eta \nabla_{\mathbf{u}} L$
    $\mathbf{v} \leftarrow \mathbf{v} + \eta \nabla_{\mathbf{v}} L$
    /* lr/update decay, max player */
    $\eta \leftarrow \gamma \times \eta$
    $\tau \leftarrow \xi \times \tau$
  end
  /* optimize Π_θ, min player */
  $\theta \leftarrow f(\theta, \nabla_\theta L)$
  /* increment iteration counter */
  $t \leftarrow t + 1$
end

number of iterations, and therefore approximate inner optimization loops are being used. To find the right balance between constraint satisfaction and policy improvement for the minimax algorithm, it is necessary to carefully adjust multiple hyperparameters. Note that an extensive hyperparameter search is undesirable in many real-world large-scale scenarios that require frequent model updates as it entails not only significant compute costs associated with the search but also increases the turn-around time to deploy refreshed models. To mitigate this issue, we suggest a meta-gradient optimization idea that adapts $\mathbf{u}$ and $\mathbf{v}$ based on a meta objective within the training process.

Specifically, we define the following meta objective:

$$L_{meta} = \mathbb{E}_{\mathbf{x},a,r,k\sim\mathbb{D}} \; [(1-\lambda) L_{\Pi_\theta}(\mathbf{x}, a, r) + \lambda \frac{\max(0, c_k^{min} - \mathcal{R}_\theta(\mathbf{x})) + \max(0, \mathcal{R}_\theta(\mathbf{x}) - c_k^{max})}{p(k)}] \,, \tag{6}$$

where λ is a hyperparameter to balance between the bandit objective and the constraint penalty terms. The second term is the macro average of constraint violation terms, in which $p(k)$ is the prior probability of samples belonging to domain k that can be easily pre-computed for a large batch of samples.

Note that (6) is not directly dependent on $\mathbf{u}$ and $\mathbf{v}$, instead we rely on online cross-validation [24, 25] to update these variables. We define an inner objective the same as the min optimization problem of (5), do a differentiable optimization step, evaluate the meta objective on another batch of data, then update $\mathbf{u}$ and $\mathbf{v}$ by taking the derivative of the meta objective through the inner optimization trace.

Algorithm 2 presents an outline of the meta gradient optimization method. Due to practical issues of dealing with high-order gradients, we only consider the immediate impact of a single inner loop update on the meta objective. We found that discarding the vanilla gradient descent used for the inner optimization and using a more advanced optimizer (e.g., Adam) to update Π_θ works best. Regarding the λ hyperparameter, we found that simply setting $\lambda = 1$ works well in practice. It effectively means that the meta-gradient solution does not require any hyperparameter adjustments (experimental evidence presented in Section 4.4).

Algorithm 2: Meta gradient constrained bandit optimization algorithm

input : $\mathbb{D}$ (dataset), η (learning rate), λ (penalty weight)

$\mathbf{u}, \mathbf{v} \leftarrow 0$
Initialize(Π_θ)
for $\mathbf{x}, a, r, k \sim \mathbb{D}$ *and* $\mathbf{x}', a', r', k' \sim \mathbb{D}$ **do**
  /* clone policy parameters */
  $\theta' \leftarrow clone(\theta)$
  /* compute inner loss with θ' */
  $L_{inner} \leftarrow Loss_{inner}(\mathbf{x}, a, r, k, \theta', \mathbf{u}, \mathbf{v})$
  /* gradient descent on cloned model */
  $\theta' \leftarrow \theta' - \eta \nabla_{\theta'} L_{inner}$
  /* compute meta loss */
  $L_{meta} \leftarrow Loss_{meta}(\mathbf{x}', a', r', k', \theta', \lambda)$
  /* diff. through inner update */
  Compute $\nabla_{\mathbf{u}} L_{meta}$ and $\nabla_{\mathbf{v}} L_{meta}$
  /* optimize $\mathbf{u}, \mathbf{v}$ using any optimizer */
  $\mathbf{u} \leftarrow f(\mathbf{u}, \nabla_{\mathbf{u}} L_{meta})$
  $\mathbf{v} \leftarrow f(\mathbf{v}, \nabla_{\mathbf{v}} L_{meta})$
  /* compute inner loss with θ */
  $L \leftarrow Loss_{inner}(\mathbf{x}, a, r, k, \theta, \mathbf{u}, \mathbf{v})$
  /* optimize Π_θ using any optimizer */
  $\theta \leftarrow f(\theta, \nabla_\theta L)$
end

Intuitively, at each training iteration, the inner objective naturally minimizes the bandit loss that is penalized by constraint violation terms proportional to the current $\mathbf{u}/\mathbf{v}$. Then, the meta objective computes a validation loss

that measures the impact of the inner policy update and $\mathbf{u}/\mathbf{v}$ on the macro-averaged constraint violations. Finally, by computing the meta-gradient of the meta objective through the inner optimization loop, $\mathbf{u}$ and $\mathbf{v}$ are updated to better encourage the constraint satisfaction for the next policy update iteration.

4. Experiments

4.1. Setup

We conduct experiments on a bandit agent for the problem of skill routing in commercial conversational systems such as Apple Siri, Amazon Alexa, Google Assistant, and Microsoft Cortana. In these systems, skill routing is a key component that takes the user's utterance as well as signals from automated speech recognition (ASR) and natural language understanding (NLU), then it decides which skill and NLU interpretation to be used to serve the request [27].

The skill routing problem in a commercial dialogue agent is a natural fit for this study as any policy change directly impacts the user experience, making safe and controlled policy updates crucial. Additionally, constraints can be defined based on the NLU domain to ensure policy safety for business-critical domains such as shopping, while potentially exploring others such as entertainment more aggressively.

Figure 1 shows an overview of the model architecture used in our experiments. Input to the model is a set of routing candidates i.e., a combination of embedded ASR, NLU, and context vectors as well as skill embeddings. The output is the softmax-normalized propensity of selecting each candidate to handle the user request. The final model has about 12M trainable parameters consisting of a language model to encode utterance, embeddings for contextual signals, and fully-connected layers. As our architecture closely follows the design choices from Park et al. [28], we refer readers to that paper for details.

To train and evaluate our models, we use logged policy actions from a current production policy. The observed reward is based on a curated function of user satisfaction metrics (refer to [29] for an example). Our dataset consists of about 40M samples divided into 85% training, 10% validation, and 5% test sets covering 27 domains that are imbalanced in the number of samples. All data used in this work was deidentified to comply with our customer privacy guidelines.

4.2. Benchmarks

In our experiments, we use three different benchmarks for the constraint settings: `global`, `critical`, and `exploration`. The `global` benchmark aims to constrain the new policy to be within an exploration limit

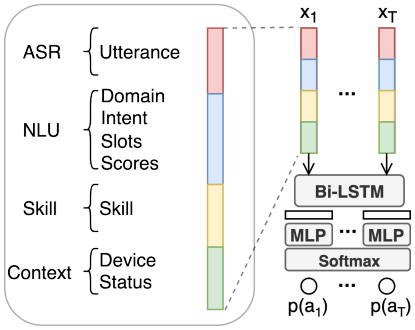

Figure 1: An overview of the network architecture: a set of hypothesis are encoded as vectors and fed to a bi-directional LSTM which is followed by a shared MLP and a softmax layer to normalize the candidate selection probabilities.

```
- description: "HomeAutomation is critical, explore less"
  domain: "HomeAutomation"
  minimum: 0.99
  maximum: 1.00
- description: "Shopping is critical, explore less"
  domain: "Shopping"
  minimum: 0.99
  maximum: 1.00
- description: "Notifications is critical, explore less"
  domain: "Notifications"
  minimum: 0.99
  maximum: 1.00
- description: "explore Knowledge"
  domain: "Knowledge"
  minimum: 0.80
  maximum: 0.95
- description: "explore Music"
  domain: "Music"
  minimum: 0.90
  maximum: 0.97
- description: "constraint for all other cases"
  domain: "*"
  minimum: 0.98
  maximum: 1.00
```

Figure 2: The constraint configuration list for the `exploration` benchmark.

for all domains. In addition to the global constraint, `critical` assert stronger limits for a set of critical domains defined based on the expert knowledge. The `exploration` benchmark extends the critical benchmark by adding constraints to encourage exploration for domains that may benefit from additional exploration. Each benchmark is a list of constraints consisting of a short description, applicable domain, and the desired replication range. Figure 2 shows the `exploration` benchmark as an example. We provide the exact constraint configurations in the appendix.

4.3. Baselines and Metrics

As the first baseline, we consider the vanilla IPS objective which disregards the constraints. Additionally, we build on the IPS baseline to consider the constraints using constraint optimization approaches: quadratic (uniform constant penalty weight), minimax (Algorithm 1), and

meta-gradient (Algorithm 2). Unless expressed otherwise, we use Adam optimizer with the default configuration [30] (denoted by f in Section 3).

Regarding hyperparameters, for the penalty weight of the quadratic method we use values from $\{0.1, 1, 10, 100, 1000\}$. For the minimax method (Algorithm 1), we found that setting τ and ξ to one while adjusting η and γ presents very similar results to adjusting all four hyperparameters. Consequently, we use a grid search of $\eta \in \{1, 0.1, 0.01\}$ and $\gamma \in \{1, 0.999, 0.995\}$ to find the best settings for each benchmark. For the meta-gradient method (Algorithm 2), we found that simply using λ equal to one in the meta objective (i.e., meta objective only focusing on the constraints) outperforms other works (see evidence in Section 4.4). As a result, it does not require adjusting any hyperparameter and the same setting is used across all benchmarks. We provide the hyperparameters used for each benchmark and method in the appendix.

Regarding the evaluation metrics, we use the expected off-policy reward as well as the rate of constraint violations averaged over all samples (micro-averaged) and individual domains (macro-averaged). To comply with our privacy and business guidelines, in all instances, we only report relative and normalized results which do not directly represent the actual scales or metric values.

We train each model until convergence or reaching 32 epochs and take the model best performing based on the macro-averaged violation rate. Each experiment was run four times using different random seeds for data sampling and weight initialization to report the mean and standard deviation of each result. We used a cluster of 32 NVIDIA V100 GPUs to process a mini-batch size of 32K samples. Each individual run took between 4 to 24 hours.

4.4. Results

Table 1 shows a comparison of results for the IPS, quadratic, minimax, and meta-gradient methods on all benchmarks. For each case, we report the expected reward and the percentage of reduction in the rate of violations compared to the simple IPS objective. The meta-gradient approach consistently shows the best results across all benchmarks. The simple quadratic method behaves very competitively to minimax, except for the `explore` benchmark which requires a more fine-grained control on multiple constraints (see Figure 2). The meta-gradient method, while having the highest reduction in constraints violations, also has very competitive performance in terms of the reward metric.

To study the impact of hyperparameters, we conducted an experiment using the `critical` benchmark by training minimax and meta-gradient based models using different hyperparameter values. Specifically, we train minimax models (Algorithm 1) using

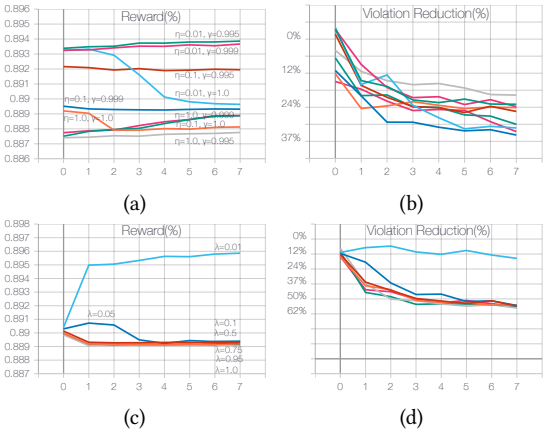

Figure 3: Comparing the hyper-parameter sensitivity for the minimax and meta-gradient methods on the `critical` benchmark. For the minimax method: (a) reward and (b) macro violation reduction wrt. different η and γ settings. For the meta-gradient method: (c) reward and (d) macro violation reduction wrt. different λ settings.

$\eta \in \{1.0, 0.1, 0.01\}$ and $\gamma \in \{1.0, 0.999, 0.995\}$. For the meta-gradient method (Algorithm 2), we use $\lambda \in \{0.01, 0.05, 0.1, 0.5, 0.75, 0.95, 1.0\}$. Figure 3 shows the results of such experiment. Based on this experiment, the minimax approach shows a much higher sensitivity to its two hyperparameters, showing a significant impact on both the reward and violation reduction metrics. However, the meta-gradient method shows much less sensitivity to the λ hyperparameter. We found that simply setting $\lambda = 1$ works very well in practice. It can be very desirable for real-world large-scale settings such as conversational systems which require frequent model updates as new features are on-boarded every day, and having a dependency on an extensive hyperparameter search is very costly, if not impractical.

To dive deeper into the reason behind the better performance for the meta-gradient algorithm compared to the minimax approach, we investigated the constraint penalty weight value for the first 3,000 iterations of training using the `global` benchmark. From Figure 4, we can see the minimax method is monotonically increasing the penalty weight with each iteration which is a behavior consistent with the gradient ascent update rule in Algorithm 1. In other words, as long as there are any constraint violations, minimax will keep increasing the penalty, which in our opinion is the reason for high sensitivity to the hyperparameters. On the other hand, the meta-gradient approach is using a validation signal to dynamically adjust the penalty weight. Consequently, it may keep the penalty term near zero for an initial phase, rapidly increase it, then decay when violations are reduced and getting a higher reward is preferred.

Method	Benchmark								
	global			critical			explore		
	reward	violation reduction		reward	violation reduction		reward	violation reduction	
	(%)	macro (%)	micro (%)	(%)	macro (%)	micro (%)	(%)	macro (%)	micro (%)
IPS	89.45±0.01	0	0	89.45±0.01	0	0	89.45±0.01	0	0
Quadratic	88.95±0.01	63.67±0.46	63.67±0.46	88.94±0.01	50.13±0.90	69.29±0.67	88.36±0.04	28.37±4.62	65.24±2.30
Minimax	88.91±0.01	63.28±0.08	63.28±0.08	88.93±0.01	37.88±0.49	62.51±0.21	88.11±0.01	61.51±0.59	81.50±0.24
MetaGrad	88.94±0.01	**75.91**±0.49	**75.91**±0.49	88.94±0.01	**60.63**±0.95	**79.69**±0.85	88.41±0.01	**78.23**±0.17	**89.95**±0.20

Table 1
A comparison of the baseline IPS method with the quadratic, minimax, and meta-gradient constrained optimization methods on different benchmarks. We report the normalized percentage of reduction in the number of constraint violations compared to the IPS method.

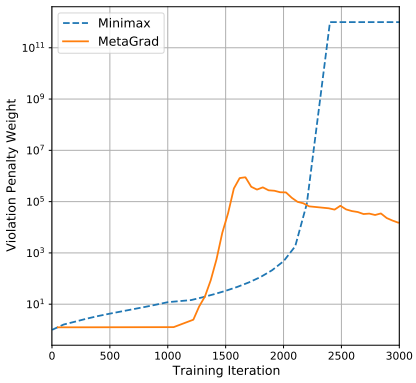

Figure 4: The constraint penalty weight values for the first 3,000 iterations of training using the `global` benchmark.

5. Conclusion

This work studied the problem of controlled exploration for off-policy contextual bandit learning. We presented a constraint optimization formulation that enables a human expert to define the boundary of the desired exploration rate for individual domains. We proposed a scalable and practical solution based on meta-gradient learning which provides the highest constraint satisfaction rates without any need for an extensive hyperparameter adjustment. Finally, we conducted experiments using data from a real-world conversation system for the skill routing problem on a set of different realistic constraint benchmarks. Based on the experimental results, we believe that the suggested controlled bandit learning approach is very promising for application in real-world bandits in which frequent safe policy updates are of paramount importance.

References

[1] A. Swaminathan, A. Krishnamurthy, A. Agarwal, M. Dudík, J. Langford, D. Jose, I. Zitouni, Off-policy evaluation for slate recommendation, arXiv preprint arXiv:1605.04812 (2016).

[2] T. Joachims, A. Swaminathan, M. de Rijke, Deep learning with logged bandit feedback, in: International Conference on Learning Representations, 2018.

[3] R. Lopez, I. S. Dhillon, M. I. Jordan, Learning from extreme bandit feedback, Proc. Association for the Advancement of Artificial Intelligence (2021).

[4] R. Jagerman, I. Markov, M. D. Rijke, Safe exploration for optimizing contextual bandits, ACM Transactions on Information Systems (TOIS) 38 (2020) 1–23.

[5] S. Daulton, S. Singh, V. Avadhanula, D. Dimmery, E. Bakshy, Thompson sampling for contextual bandit problems with auxiliary safety constraints, arXiv preprint arXiv:1911.00638 (2019).

[6] S. Amani, M. Alizadeh, C. Thrampoulidis, Linear stochastic bandits under safety constraints, arXiv preprint arXiv:1908.05814 (2019).

[7] M. Hoffman, B. Shahriari, N. Freitas, On correlation and budget constraints in model-based bandit optimization with application to automatic machine learning, in: Artificial Intelligence and Statistics, PMLR, 2014, pp. 365–374.

[8] S. Guha, K. Munagala, Multi-armed bandits with limited exploration, in: Proceedings of the Annual Symposium on Theory of Computing (STOC), Citeseer, 2007.

[9] C. Riquelme, G. Tucker, J. Snoek, Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling, arXiv preprint arXiv:1802.09127 (2018).

[10] M. Collier, H. U. Llorens, Deep contextual multi-armed bandits, arXiv preprint arXiv:1807.09809 (2018).

[11] A. Balakrishnan, D. Bouneffouf, N. Mattei, F. Rossi, Using contextual bandits with behavioral constraints for constrained online movie recommendation., in: IJCAI, 2018, pp. 5802–5804.

[12] J. Liu, J. Ye, Efficient euclidean projections in linear

time, in: Proceedings of the 26th Annual International Conference on Machine Learning, 2009, pp. 657–664.

[13] P. L. Donti, D. Rolnick, J. Z. Kolter, Dc3: A learning method for optimization with hard constraints, International Conference on Learning Representations (ICLR) (2021).

[14] Y. Nandwani, A. Pathak, P. Singla, et al., A primal dual formulation for deep learning with constraints, in: Advances in Neural Information Processing Systems, 2019, pp. 12157–12168.

[15] H. Kervadec, J. Dolz, J. Yuan, C. Desrosiers, E. Granger, I. B. Ayed, Constrained deep networks: Lagrangian optimization via log-barrier extensions, arXiv:1904.04205 (2019).

[16] N. Andrei, Penalty and augmented lagrangian methods, in: Continuous Nonlinear Optimization for Engineering Applications in GAMS Technology, Springer, 2017, pp. 185–201.

[17] R. Glowinski, P. Le Tallec, Augmented Lagrangian and operator-splitting methods in nonlinear mechanics, SIAM, 1989.

[18] C. Jin, P. Netrapalli, M. I. Jordan, Minmax optimization: Stable limit points of gradient descent ascent are locally optimal, arXiv preprint arXiv:1902.00618 (2019).

[19] Z. Xu, H. van Hasselt, M. Hessel, J. Oh, S. Singh, D. Silver, Meta-gradient reinforcement learning with an objective discovered online, arXiv preprint arXiv:2007.08433 (2020).

[20] L. Kirsch, S. van Steenkiste, J. Schmidhuber, Improving generalization in meta reinforcement learning using learned objectives, in: International Conference on Learning Representations, 2019.

[21] S. Ritter, J. Wang, Z. Kurth-Nelson, S. Jayakumar, C. Blundell, R. Pascanu, M. Botvinick, Been there, done that: Meta-learning with episodic recall, in: International Conference on Machine Learning, PMLR, 2018, pp. 4354–4363.

[22] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 1126–1135.

[23] M. Ren, W. Zeng, B. Yang, R. Urtasun, Learning to reweight examples for robust deep learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 4334–4343.

[24] R. S. Sutton, Adapting bias by gradient descent: An incremental version of delta-bar-delta, in: AAAI, San Jose, CA, 1992, pp. 171–176.

[25] Z. Xu, H. van Hasselt, D. Silver, Meta-gradient reinforcement learning, arXiv preprint arXiv:1805.09801 (2018).

[26] A. Nichol, J. Achiam, J. Schulman, On first-order meta-learning algorithms, arXiv preprint arXiv:1803.02999 (2018).

[27] R. Sarikaya, The technology behind personal digital assistants: An overview of the system architecture and key components, IEEE Signal Processing Magazine 34 (2017) 67–81.

[28] S. Park, H. Li, A. Patel, S. Mudgal, S. Lee, Y.-B. Kim, S. Matsoukas, R. Sarikaya, A scalable framework for learning from implicit user feedback to improve natural language understanding in large-scale conversational ai systems, arXiv preprint arXiv:2010.12251 (2020).

[29] M. Kachuee, H. Yuan, Y.-B. Kim, S. Lee, Self-supervised contrastive learning for efficient user satisfaction prediction in conversational agents, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021, pp. 4053–4064.

[30] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

A. Appendix

A.1. Constraint Benchmarks

Figure 5 presents the definition of constraint benchmarks used in this paper: `global`, `critical`, and `explore`. The `global` benchmark sets a general minimum replication rate for all domains. The `critical` benchmark defines a tighter minimum replication rate for three business-critical domains (home automation, shopping, and notifications) and a more relaxed default case for all other domains. In the `explore` benchmark, we extend the `critical` benchmark to include exploration encouragement for the knowledge and music domains.

```yaml
- description: "constraint for all cases"
  domain: "*"
  minimum: 0.99
  maximum: 1.00
```

(a) `global` benchmark

```yaml
- description: "domain HomeAutomation is critical, explore less"
  domain: "HomeAutomation"
  minimum: 0.99
  maximum: 1.00
- description: "Shopping is critical, explore less"
  domain: "Shopping"
  minimum: 0.99
  maximum: 1.00
- description: "Notifications is critical, explore less"
  domain: "Notifications"
  minimum: 0.99
  maximum: 1.00
- description: "constraint for all other cases"
  domain: "*"
  minimum: 0.98
  maximum: 1.00
```

(b) `critical` benchmark

```yaml
- description: "HomeAutomation is critical, explore less"
  domain: "HomeAutomation"
  minimum: 0.99
  maximum: 1.00
- description: "Shopping is critical, explore less"
  domain: "Shopping"
  minimum: 0.99
  maximum: 1.00
- description: "Notifications is critical, explore less"
  domain: "Notifications"
  minimum: 0.99
  maximum: 1.00
- description: "explore Knowledge"
  domain: "Knowledge"
  minimum: 0.80
  maximum: 0.95
- description: "explore Music"
  domain: "Music"
  minimum: 0.90
  maximum: 0.97
- description: "constraint for all other cases"
  domain: "*"
  minimum: 0.98
  maximum: 1.00
```

(c) `explore` benchmark

Figure 5: The constraint benchmarks used in this paper: (a) `global`, (b) `critical`, and (c) `explore`.

A.2. Selected Hyperparameters

Table 2 shows the final selected hyperparameters for each benchmark and method. The definition of each hyperparameter is presented in Algorithm 1 and 2.

		Benchmark		
Method		`global`	`critical`	`explore`
Quadratic	w	10	1000	1000
Minimax	η	0.1	0.1	1
	γ	1	0.999	1
Meta-Grad	λ	1	1	1

Table 2
The selected hyperparameters for each benchmark and method.