Constrained Policy Optimization for Controlled Self-Learning in Conversational AI Systems

Mohammad Kachuee, Sungjin Lee
Amazon Alexa AI
{kachum, sungjinl}@amazon.com

Abstract

Recently, self-learning methods based on user satisfaction metrics and contextual bandits have shown promising results to enable consistent improvements in conversational AI systems. However, directly targeting such metrics by off-policy bandit learning objectives often increases the risk of making abrupt policy changes that break the current user experience. In this study, we introduce a scalable framework for supporting fine-grained exploration targets for individual domains via user-defined constraints. For example, 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. We present a novel meta-gradient learning approach 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 conducted extensive experiments on a real-world conversational AI and using a set of realistic constraint benchmarks. The proposed approach has been deployed in production for a large-scale commercial assistant, enabling the best balance between the policy value and constraint satisfaction rate.

1 Introduction

Conversational AI systems such as Apple Siri, Amazon Alexa, Google Assistant, and Microsoft Cortana rely on multiple processing components for speech recognition, natural language understanding (NLU), taking proper actions, and generating a response to the user. In such a system, a skill routing block is responsible for selecting the right skill and NLU interpretation to serve the request. Skill routing is a challenging problem as thousands of skills are present in real-world conversational systems and new skills are being introduced every day. In such scenario, gathering human annotations is very expensive and suffers from high turn-around times. Moreover, often more than one skill is capable of serving a request which makes human supervision even more challenging due to the lack of clear ground-truth assignments (Sarikaya, 2017).

Recently, self-learning methods have been proposed that leverage customer experience signals to define reward values and create a closed feedback loop (Karampatziakis et al., 2019). In contrast to more traditional methods that are based on replication of rule-based systems or defect relabeling (Park et al., 2020), self-learning methods continuously explore different routing alternatives and leverage user feedback to improve their decisions (Kachuee et al., 2022).

Despite their scalability and efficiency, because self-learning approaches directly optimize routing decisions to achieve highest rewards, they suffer from instability issues impacting the user experience. Specifically, off-policy contextual bandits frequently used as the policy learning algorithm are susceptible to off-policy optimization errors, resulting in potentially breaking the current user experience due to overestimation of action values or excessive explorations (Swaminathan et al., 2016; Joachims et al., 2018; Lopez et al., 2021). Such instabilities and drastic changes in the agent's behavior not only regress user retention and trust, but also manifest as direct revenue loss for business-critical domains such as shopping.

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 (Jagerman et al., 2020; Daulton et al., 2019; Amani et al., 2019) and budgeted bandit learning (Hoffman et al., 2014; Guha and Munagala, 2007), 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 be-

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 5: Industry Track, pages 43–52

havior 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. Note that in a task-oriented conversational agent, domains are typically defined based on NLU interpretation of the request (e.g. music, shopping, books).

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 and practical approach. The main contributions of this paper are as follows: (i) 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. (ii) Presenting a solution based on the primal-dual minimax constraint optimization method that is effective but requires adjusting a few hyperparameters. (iii) 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. (iv) Conducting extensive online and offline 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 Skill Routing in Conversational AIs

In contrast to traditional rule-based systems, model-based skill routing approaches leverage machine learning models to understand a user request and predict the best action to serve the request (Li et al., 2021; Park et al., 2020).

To improve scalability, self-learning methods have been proposed that rely on user feedback rather than human annotations to learn and improve their skill routing policies in a closed-loop. The recent work by Kachuee et al. (2022) is an excellent example of such approach in which model-based customer satisfaction metrics (Kachuee et al., 2021) are used to define the reward function, then a stochastic mixture of replication and bandit models is used to control the exploration rate and safeguard the user experience. Nonetheless, such design may result in sub-optimal decisions as the bandit optimization does not consider the exploration budgets, the stochastic mixture may not be sufficiently fine-grained to protect user experience in smaller traffic segments, and deploying such architecture requires dealing with additional complexity of maintaining a separate replication model.

2.2 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 (Guha and Munagala, 2007) and automated machine learning (Hoffman et al., 2014). Hoffman et al. (2014) 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. (2019) 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, constraints can be defined in the action space or in terms of model updates. For example, Daulton et al. (2019) solves a two-metric setting in which the reward is being maximized while enforcing a limit for regression on an auxiliary metric compared to a baseline status quo model. Balakrishnan et al. (2018) attempts to learn behavioral constraints by balancing between replication of the current baseline policy and making new actions that show promising rewards. In (Jagerman et al., 2020) 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.

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 $k \in 1 \dots M$ 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 $\mathbf{x}$:

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

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 robust and controlled 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:

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

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:

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

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:

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

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. In this paper, to solve this problem, we use the primal-dual minimax method suggested by Nandwani et al. (2019) (Section 3.2) as well as a novel meta-learning method (Section 3.3).

3.2 Minimax Primal-Dual Method

Nandwani et al. (2019) 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:

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

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 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.

Algorithm 1: Minimax constrained bandit

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($\Pi_\theta$)

for $\mathbf{x}, a, r, k \sim D$ **do**

  /* 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 */

    $\eta \leftarrow \gamma \times \eta$

    $\tau \leftarrow \xi \times \tau$

  end

  /* optimize Π_θ, min player */

  $\theta \leftarrow f(\theta, \nabla_\theta L)$

  /* increment counter */

  $t \leftarrow t + 1$

end

3.3 Meta Gradient Method

Theoretically, the primal-dual minimax method is capable of achieving Pareto optimal solutions (Jin et al., 2019; Nandwani et al., 2019). However, in practice, it is infeasible to train for an infinite 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 scenarios 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 the meta objective as:

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

where λ is a hyperparameter to balance between the bandit objective and the constraint penalty terms. The second term is the macro average of violation penalties, 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 (Sutton, 1992; Xu et al., 2018) 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-grad constrained bandit

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

$\mathbf{u}, \mathbf{v} \leftarrow 0$; Initialize($\Pi_\theta$)

for $\mathbf{x}, a, r, k \sim \mathbb{D}$ *and* $\mathbf{x}', a', r', k' \sim \mathbb{D}$ **do**

  /* clone parameters */

  $\theta' \leftarrow clone(\theta)$

  /* inner loss with θ' */

  $L_{inner} \leftarrow Loss_{inner}(\mathbf{x}, a, r, k, \theta', \mathbf{u}, \mathbf{v})$

  /* grad. 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 $\nabla_{\mathbf{v}} L_{meta}$

  /* use any optimizer for $\mathbf{u}, \mathbf{v}$ */

  $\mathbf{u} \leftarrow f(\mathbf{u}, \nabla_{\mathbf{u}} L_{meta})$

  $\mathbf{v} \leftarrow f(\mathbf{v}, \nabla_{\mathbf{v}} L_{meta})$

  /* inner loss with θ */

  $L \leftarrow Loss_{inner}(\mathbf{x}, a, r, k, \theta, \mathbf{u}, \mathbf{v})$

  /* use any optimizer for Π_θ */

  $\theta \leftarrow f(\theta, \nabla_\theta L)$

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. Thanks to the online cross-validation update for $\mathbf{u}$ and $\mathbf{v}$, the

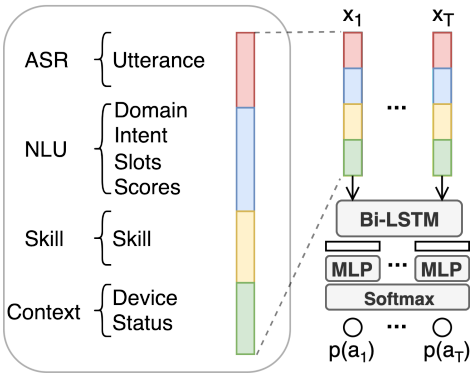

Figure 1: Model architecture overview: a set of hypothesis are encoded as vectors and fed to a bi-directional LSTM followed by a shared MLP and a softmax layer to get the candidate selection probabilities .

meta-gradient method adjusts the penalty weights such that their value does not unnecessarily keep increasing when it does not result in further improvements to the constraint satisfaction.

4 Experiments

4.1 Setup

In a commercial dialogue agent, making controlled policy updates is crucial because any change in the skill routing policy directly impacts the user experience. Making abrupt policy changes may negatively impact user retention and in certain business-critical domains may result in loss of revenue.

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 Kachuee et al. (2022), we refer interested readers to that paper for details.

To train and evaluate our models, we use logged data from a current production policy. The observed reward is based on a curated function of user satisfaction metrics (Kachuee et al., 2021). Our dataset consists of about 40M samples divided into 85% training, 10% validation, and 5% test sets covering 27 domains with imbalanced number of samples. 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 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. 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 for 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 (Kingma and Ba, 2014) .

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. As a result, it does not require adjusting any hyperparameter and the same setting is used across all benchmarks. We provide additional experiment details, sensitivity analysis, and the final hyperparametersin Appendix A.2, A.3, and A.4.

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

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.

Method	reward	Violation Reduction	Replication
	(%)	(%)	(%)
RPDR	-0.01 (p>0.05)	0	98.13
MetaGrad	+0.19 (p<0.05)	38.05	99.11

Table 2: Comparison of the proposed method (MetaGrad) and the robust self-learning method by Kachuee et al. (2022) (RPDR) using an online A/B experiment. We report: percentage of change in the reward compared to a control model, violation reduction for the MetaGrad normalized by the RPDR result, and percentage of replication compared to the control policy actions.

4.4 Results

4.4.1 Offline Experimentation

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 . The meta-gradient method, while having the highest reduction in constraints violations, also has very competitive performance in terms of the reward metric.

4.4.2 Online Experimentation

We conducted an A/B experiment to compare the proposed method with the stochastic gating method of Kachuee et al. (2022) for robust self-learning (indicated by RPDR in the table). We conducted our A/B in two phases, deploying and comparing each approach to a baseline skill routing production system. Each phase took one week and consisted of traffic from about 6M customers (3M control and 3M treatment). For the RPDR method, we used a target replication rate of 99% for each domain. The meta-gradient model was trained with the `global` benchmark, constraining to a similar 99% replication. For both RPDR and MetaGrad models, we used the same training set which was collected from the control model behavior and followed the same model architecture.

Table 2 presents the results of the A/B experiment. For each method we report the percentage of changes in the achieved reward compared to the control model. For violation reduction, we report the percentage of reduction for MetaGrad compared to the RPDR method. For the replication metric, we simply report the percentage of time that each policy makes actions that replicate the control model decision. As we can see from the results, MetaGrad approach not only shows more stable behavior by better constraint satisfaction and replication rates, but it also achieves statistically significant improvements in the reward value.

5 Conclusion

This paper studied the problem of controlled exploration to control the policy updates in self-learning skill routing systems. We presented a constrained optimization formulation that enables defining 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 extensive hyperparameter adjustment. Finally, we conducted experiments on a real-world conversation system for the skill routing problem. The proposed method was deployed in the production as it showed not only more control over policy changes but also gains in the policy value.

Limitations

While we conducted extensive experiments and demonstrated the effectiveness of the suggested approach for controlled bandit learning in the context of the skill routing problem, there are multiple directions of improvement for future studies. We believe one of the limitations of the suggested constrained optimization framework is that it relies on expert-defined conditions on an arbitrary segmentation of samples. It entails the need for human intervention and manual constraint definition/optimization which can be challenging. Another limitation we faced was during our experiments which showed additional compute overhead of between 2 to 3 times for different constrained optimization methods due to additional optimization objectives, inner loops, and backward passes.

Ethics Statement

The presented work is focused on improving robustness of off-policy bandit updates in conversational systems by introducing robustness constraints on the policy behavior. We do not believe there is any additional risk associated with this work when using the suggested platform on constraints that encourage controlled deviations from a current baseline. Regarding human data handling practices, we ensured anonymity of data samples used in this study and did not reveal any specifics that would violate our internal policies or our customer privacy policies.

References

Sanae Amani, Mahnoosh Alizadeh, and Christos Thrampoulidis. 2019. Linear stochastic bandits under safety constraints. *arXiv preprint arXiv:1908.05814*.

Avinash Balakrishnan, Djallel Bouneffouf, Nicholas Mattei, and Francesca Rossi. 2018. Using contextual bandits with behavioral constraints for constrained online movie recommendation. In *IJCAI*, pages 5802–5804.

Samuel Daulton, Shaun Singh, Vashist Avadhanula, Drew Dimmery, and Eytan Bakshy. 2019. Thompson sampling for contextual bandit problems with auxiliary safety constraints. *arXiv preprint arXiv:1911.00638*.

Sudipto Guha and Kamesh Munagala. 2007. Multi-armed bandits with limited exploration. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*. Citeseer.

Matthew Hoffman, Bobak Shahriari, and Nando Freitas. 2014. On correlation and budget constraints in model-based bandit optimization with application to automatic machine learning. In *Artificial Intelligence and Statistics*, pages 365–374. PMLR.

Rolf Jagerman, Ilya Markov, and Maarten De Rijke. 2020. Safe exploration for optimizing contextual bandits. *ACM Transactions on Information Systems (TOIS)*, 38(3):1–23.

Chi Jin, Praneeth Netrapalli, and Michael I Jordan. 2019. Minmax optimization: Stable limit points of gradient descent ascent are locally optimal. *arXiv preprint arXiv:1902.00618*.

Thorsten Joachims, Adith Swaminathan, and Maarten de Rijke. 2018. Deep learning with logged bandit feedback. In *International Conference on Learning Representations*.

Mohammad Kachuee, Jinseok Nam, Sarthak Ahuja, Jin-Myung Won, and Sungjin Lee. 2022. Scalable and robust self-learning for skill routing in large-scale conversational ai systems. *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Mohammad Kachuee, Hao Yuan, Young-Bum Kim, and Sungjin Lee. 2021. 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*, pages 4053–4064.

Nikos Karampatziakis, Sebastian Kochman, Jade Huang, Paul Mineiro, Kathy Osborne, and Weizhu Chen. 2019. Lessons from contextual bandit learning in a customer support bot. *arXiv preprint arXiv:1905.02219*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Han Li, Sunghyun Park, Aswarth Dara, Jinseok Nam, Sungjin Lee, Young-Bum Kim, Spyros Matsoukas, and Ruhi Sarikaya. 2021. Neural model robustness for skill routing in large-scale conversational ai systems: A design choice exploration. *arXiv preprint arXiv:2103.03373*.

Romain Lopez, Inderjit S Dhillon, and Michael I Jordan. 2021. Learning from extreme bandit feedback. *Proc. Association for the Advancement of Artificial Intelligence*.

Yatin Nandwani, Abhishek Pathak, Parag Singla, et al. 2019. A primal dual formulation for deep learning with constraints. In *Advances in Neural Information Processing Systems*, pages 12157–12168.

Sunghyun Park, Han Li, Ameen Patel, Sidharth Mudgal, Sungjin Lee, Young-Bum Kim, Spyros Matsoukas, and Ruhi Sarikaya. 2020. 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*.

Ruhi Sarikaya. 2017. The technology behind personal digital assistants: An overview of the system architecture and key components. *IEEE Signal Processing Magazine*, 34(1):67–81.

Richard S Sutton. 1992. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *AAAI*, pages 171–176. San Jose, CA.

Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miroslav Dudík, John Langford, Damien Jose, and Imed Zitouni. 2016. Off-policy evaluation for slate recommendation. *arXiv preprint arXiv:1605.04812*.

Zhongwen Xu, Hado van Hasselt, and David Silver. 2018. Meta-gradient reinforcement learning. *arXiv preprint arXiv:1805.09801*.

A Appendix

A.1 Constraint Benchmarks

Figure 2 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 2: The constraint benchmarks used in this paper: (a) `global`, (b) `critical`, and (c) `explore`.

A.2 Training Details

We train each model until convergence or reaching 32 epochs and take the best performing model based on the macro-averaged violation rate measured on the validation set. 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.

A.3 Selected Hyperparameters

Table 3 shows the final selected hyperparameters for each benchmark and method. The definition of each hyper-parameter 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 3: The selected hyperparameters for each benchmark and method.

A.4 Impact of Hyperparameters

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 $\eta \in \{1.0, 0.1, 0.01\}$ 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.

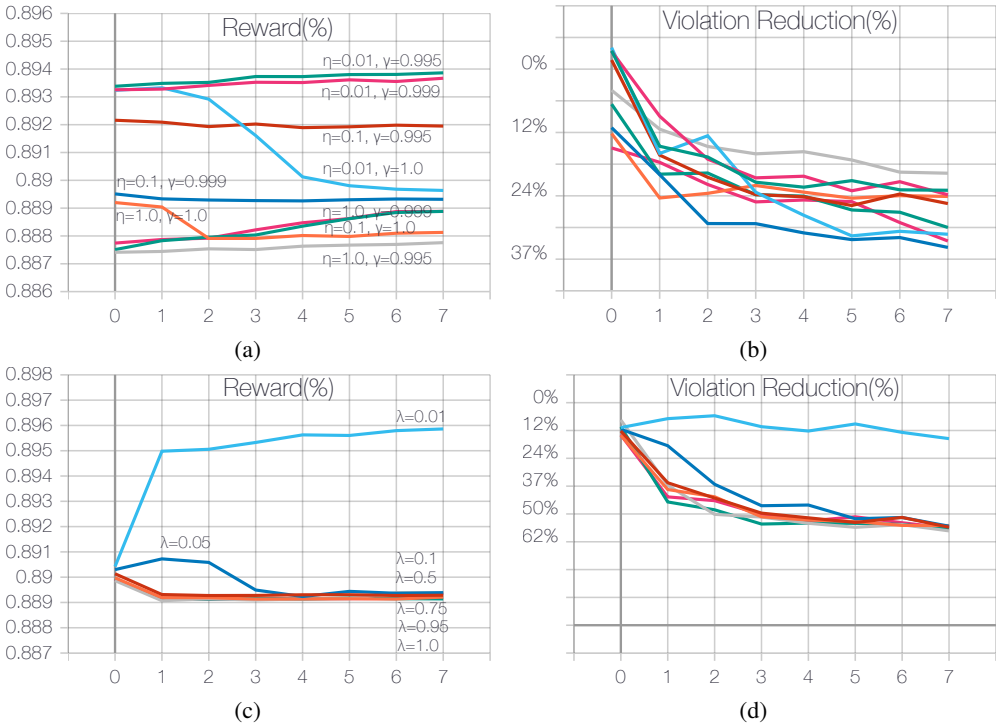

Figure 3: Comparing the hyper-parameter sensitivity for the minimax and meta-gradient methods on the `critical` benchmark. For the minimax method: (a) reward and (b) macro violation reduction wrt. different η and γ settings. For the meta-gradient method: (c) reward and (d) macro violation reduction wrt.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

A.5 Analysis of Penalty Weights

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.

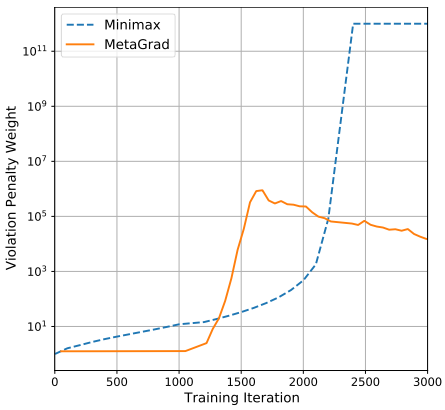

Figure 4: The constraint penalty weight values for the first 3,000 iterations of training using the `global` benchmark.