Off-policy Bandits with Deficient Support

Noveen Sachdeva*†
International Institute of Information Technology
Hyderabad, India
ernoveen@gmail.com

Yi Su†
Cornell University
Ithaca, NY, USA
ys756@cornell.edu

Thorsten Joachims
Cornell University
Ithaca, NY, USA
tj@cs.cornell.edu

ABSTRACT

Learning effective contextual-bandit policies from past actions of a deployed system is highly desirable in many settings (e.g. voice assistants, recommendation, search), since it enables the reuse of large amounts of log data. State-of-the-art methods for such off-policy learning, however, are based on inverse propensity score (IPS) weighting. A key theoretical requirement of IPS weighting is that the policy that logged the data has "full support", which typically translates into requiring non-zero probability for any action in any context. Unfortunately, many real-world systems produce support deficient data, especially when the action space is large, and we show how existing methods can fail catastrophically. To overcome this gap between theory and applications, we identify three approaches that provide various guarantees for IPS-based learning despite the inherent limitations of support-deficient data: restricting the action space, reward extrapolation, and restricting the policy space. We systematically analyze the statistical and computational properties of these three approaches, and we empirically evaluate their effectiveness. In addition to providing the first systematic analysis of support-deficiency in contextual-bandit learning, we conclude with recommendations that provide practical guidance.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**; • **Computing methodologies → Learning from implicit feedback**.

KEYWORDS

contextual bandits, counterfactual reasoning, log data, implicit feedback, off-policy learning

ACM Reference Format:
Noveen Sachdeva, Yi Su, and Thorsten Joachims. 2020. Off-policy Bandits with Deficient Support. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403139

*Work done during internship at Cornell University.
†Equal contribution.

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403139

1 INTRODUCTION

Many interactive systems (e.g., voice assistants, recommender systems) can be modeled as *contextual bandit* problems [15]. In particular, each user request provides a context (e.g., user profile, query) for which the system selects an action (e.g., recommended product) and receives a reward (e.g., purchase, click). Such contextual-bandit data is logged in large quantities as a by-product of normal system operation [12, 17, 18], making it an attractive and low-cost source of training data. With terabytes of log data readily available in many online systems, a range of algorithms has been proposed for batch learning from such logged contextual-bandit feedback [5, 6, 20–22, 24, 26]. However, as we will argue below, these algorithms require an assumption about the log data that makes them unsuitable for many real-world applications.

This assumption is typically referred to as the positivity or support assumption, and it is required by the Empirical Risk Minimization (ERM) objective that these algorithms optimize. Specifically, unlike in online learning for contextual bandits [1, 29], batch learning from bandit feedback (BLBF) operates in the off-policy setting. During off-policy learning, the algorithm has to address the counterfactual question of how much reward each policy in the policy space would have received, if it had been used instead of the logging policy. To this effect, virtually all state-of-the-art off-policy learning methods for contextual-bandit problems rely on counterfactual estimators [3, 5, 6, 22, 24, 26] that employ inverse propensity score (IPS) weighting to get an unbiased ERM objective. Unlike regression-based direct-modeling (DM) approaches that are often hampered by bias from model misspecification, IPS allows a controllable bias-variance trade-off through clipping and other variance-regularization techniques [20, 21, 24].

Unfortunately, IPS and its guarantee of unbiasedness break down when the logging policy does not have full support – meaning that some actions have zero probability of being selected under the logging policy. In this case IPS can be highly biased. Full support is an unreasonable assumption in many real-world systems, especially when the action space is large and many actions have poor rewards. For example, in a recommender system with a large catalog (e.g. movies, music), it may be that only a small percentage of the actions have support under the logging policy. We will show that existing learning algorithms can fail catastrophically on such support deficient data.

In this paper, we explore how to deal with support deficient log data in off-policy contextual-bandit learning. Since support deficiency translates into blind spots where we do not have any knowledge about the rewards, accounting for these blind spots during training is crucial for robust learning. We characterize three

approaches for dealing with support deficiency. The first approach is to restrict the action space to those actions that have support under the logging policy. Second, we explore imputation methods that extrapolate estimated rewards to those blind spots. And, third, we restrict the policy space to only those policies that have limited exposure to the blind spots. To make the latter approach computationally tractable, we define a new measure of Support Divergence between policies, show how it can be estimated efficiently without closed-form knowledge of the logging policy, and how it can be used as a constraint on the policy space. We analyze the statistical and computational properties of all three approaches and perform an extensive empirical evaluation. We find that restricting the policy space is particularly effective, since it is computationally efficient, empirically effective at learning good policies, and convenient to use in practice.

2 RELATED WORK

The problem of learning optimal policies by re-using logged data is referred to as off-policy learning. There has been considerable interest in developing efficient off-policy learning in the contextual bandit setting. However, to the best of our knowledge, no previous work has comprehensively investigated the problem of support deficiency, which is an important and pervasive problem in many real-world systems. The prior work on off-policy learning can be classified into two fundamentally different approaches. The first – called Direct Modeling (DM) – is based on a reduction to supervised learning, where a regression estimate is trained to predict rewards [2]. To derive a policy, the action with the highest predicted reward is chosen. A drawback of this simple approach is the bias that results from misspecification of the regression model. For real-world data, due to non-linearity or partial observability of the environment, regression models are often substantially misspecified. Hence, the DM approach often does not perform well empirically.

The second approach is based on policy learning via ERM with a counterfactual risk estimator [5, 22, 24, 25]. Inverse propensity score (IPS) weighting is one of the most popular estimators to be used as empirical risk. However, policy learning algorithms based on IPS and related estimators [20, 21, 24–26] require the assumption that the logging policy has full support for every policy in the policy space. One exception is the work of [19]. They relax the assumption to the existence of an optimal policy such that the logging policy covers the support of this optimal policy. However, this is an untestable assumption that does not provide guarantees for real-world applications.

More generally, batch learning from bandit feedback can be viewed as off-policy learning in the reinforcement learning (RL) literature, which considers learning optimal policies in the sequential decision-making setting. Similar to the approaches in contextual bandit learning, RL methods cluster into two categories: (1) value function based and (2) importance sampling based. For importance weighting based methods, such as policy gradient [29], the variance of the underlying gradient grows exponentially with the horizon, making it highly undesirable. It is worth noting that this is fundamentally different for one-step contextual bandits, where importance-sampling based estimators are the most competitive ones. For value function based approaches [10, 14], the objective is based on some estimate of the value function, either through fitting an MDP model and evaluating the value function based on the estimated MDP [10] (a.k.a model-based in RL) or using bootstrapping to find the value function for the optimal policy directly, such as Q-learning [28] (a.k.a model-free in RL). However, for model-based methods, the bias problem could be severe if a wrong model class is chosen, and the bias problem is very difficult to diagnose in general. On the other hand, for model-free methods, Sutton and Barto [23] identify a deadly triad of function approximation, bootstrapping, and off-policy learning. It emphasizes that function approximation equipped with Q-learning can diverge in the off-policy learning setting, making most off-policy RL methods very conservative in extrapolation because of the severe error-propagation issue [7, 16]. In this work, we focus on the contextual bandit problem, which has many direct applications in recommender systems and online search. We also believe that for developing better off-policy estimators in RL, it is fundamental to first understand how to handle these cases in the more tractable, contextual-bandit case.

In this paper, we explore three approaches to addressing off-policy learning with support deficiency and discuss how existing approaches could be fit into this framework. First, our conservative extrapolation method is related to the method proposed by [19]. They focus on the correction of the state distribution by defining an augmented MDP, and pessimistic imputation is used to get an estimate for policy-gradient learning. Second, our method of restricting the policy space uses a surrogate for the support divergence of two policies that was previously used as control variate in the SNIPS estimator [25]. It also appeared in the Lagrangian formulation of the BanditNet objective [11] and in the gradient update of the REINFORCE algorithm [29]. This connection gives interesting new insight that the baselines used in policy-gradient algorithms not only help to reduce variance in gradients [8], but that they also connect to the problem of support deficiency in the off-policy setting.

3 OFF-POLICY LEARNING WITH DEFICIENT SUPPORT

We start by formally defining the problem of learning a contextual-bandit policy in the BLBF setting. Input to the policy are contexts $x \in \mathcal{X}$ drawn i.i.d. from a fixed but unknown distribution $P(\mathcal{X})$. Given context x, the system executes a possibly stochastic policy $\pi(\mathcal{Y}|x)$ that selects an action $y \in \mathcal{Y}$. For this context and action pair, the system observes a reward $r \in [r_{min}, r_{max}]$ from $P(r|x, y)$. Given a space of policies Π, the reward of any policy $\pi \in \Pi$ is defined as

$$R(\pi) = \mathop{\mathbb{E}}_{x\sim P(x)} \mathop{\mathbb{E}}_{y\sim\pi(y|x)} \mathop{\mathbb{E}}_{r\sim P(r|x,y)} [r]. \tag{1}$$

In the BLBF setting, the learning algorithm is given a dataset

$$\mathcal{D} := \{x_i, y_i, r_i, \pi_0(y_i|x_i)\}_{i=1}^n$$

of past system interactions which consists of context-action-reward-propensity tuples. The propensity $\pi_0(y_i|x_i)$ is the probability of selecting action y_i for context x_i under the policy π_0 that was used to log the data. We call π_0 the logging policy, and we will discuss desired conditions on the stochasticity of π_0 in the following. The

goal of off-policy learning is to exploit the information in the logged data $\mathcal{D}$ to find a policy $\hat{\pi} \in \Pi$ that has high reward $R(\hat{\pi})$.

Analogous to the ERM principle in supervised learning, off-policy learning algorithms typically optimize a counterfactual estimate $\hat{R}(\pi)$ of $R(\pi)$ as the training objective [3, 17, 18, 24].

$$\hat{\pi} = \arg\max_{\pi \in \Pi} [\hat{R}(\pi)] \tag{2}$$

For conciseness, we ignore additional regularization terms in the objective [24], since they are irrelevant to the main point of this paper. As for counterfactual estimator $\hat{R}(\pi)$, most algorithms rely on some form of IPS weighting [5, 21, 22, 24, 25, 27] to correct the distribution mismatch between the logging policy π_0 and each target policy $\pi \in \Pi$.

$$\hat{R}_{IPS}(\pi) = \frac{1}{n} \sum_{i=1}^{n} \frac{\pi(y_i|x_i)}{\pi_0(y_i|x_i)} r_i. \tag{3}$$

A crucial condition for the effectiveness of the IPS estimator (and similar estimators like SNIPS [25]) is that the logging policy π_0 assigns non-zero probability to all actions that have non-zero probability under the target policy π we aim to evaluate. This condition is known as positivity or full support, and it is defined as follows.

Definition 1 (Full support)**.** *The logging policy π_0 is said to have full support for π when $\pi_0(y|x) > 0$ for all actions $y \in \mathcal{Y}$ and contexts $x \in \mathcal{X}$ for which $\pi(y|x) > 0$.*

It is known that the IPS estimator is unbiased, $\mathbb{E}_{\mathcal{D}}[\hat{R}_{IPS}(\pi)] = R(\pi)$, if the logging policy π_0 has full support for π [18].

To ensure unbiased ERM learning, algorithms that use the IPS estimator require that the logging policy π_0 has full support for all policies $\pi \in \Pi$ in the policy space. For sufficiently rich policy spaces, like deep-networks $f_w(x, y)$ with softmax outputs of the form

$$\pi_w(y|x) = \frac{exp(f_w(x, y))}{\sum_{y' \in \mathcal{Y}} exp(f_w(x, y'))}, \tag{4}$$

this means that the logging policy π_0 needs to assign non-zero probability to every action y in every context x. This is a strong condition that is not feasible in many real-world systems, especially if the action space is large and many actions have poor reward.

If the support requirement is violated, ERM learning can fail catastrophically. We will show in the following that the underlying reason is bias, not excessive variance that could be remedied through clipping or variance regularization [21, 24]. To quantify how support deficient a logging policy is, we denote the set of unsupported actions for context x under π_0 as

$$\mathcal{U}(x, \pi_0) := \{y \in \mathcal{Y} | \pi_0(y|x) = 0\}.$$

The bias of the IPS estimator is then characterized by the expected reward on the unsupported actions.

Proposition 1. *Given contexts $x \sim P(\mathcal{X})$ and logging policy $\pi_0(\mathcal{Y}|x)$, the bias of $\hat{R}_{IPS}$ for target policy $\pi(\mathcal{Y}|x)$ is equal to the expected reward on the unsupported action sets, i.e.,*

$$bias(\hat{R}_{IPS}(\pi)) = \mathop{\mathbb{E}}_{x} \left[- \sum_{y \in \mathcal{U}(x, \pi_0)} \pi(y|x) \delta(x, y) \right].$$

The proof is provided in Appendix A.1. From Proposition 1, it is clear that support deficient log data can drastically mislead ERM learning. To quantify the effect of support deficiency on ERM, we define the support divergence between a logging policy π_0 and a target policy π as follows.

Definition 2 (Support Divergence)**.** *For contexts $x \sim P(\mathcal{X})$ and any corresponding pair of target policy π and logging policy π_0, the Support Divergence is defined as*

$$\mathcal{D}_{\mathcal{X}}(\pi|\pi_0) := \mathop{\mathbb{E}}_{x \sim P(\mathcal{X})} \left[\sum_{y \in \mathcal{U}(x, \pi_0)} \pi(y|x) \right]. \tag{5}$$

With this definition in hand, we can quantify the effect of support deficiency on ERM learning for a policy space Π under logging policy π_0.

Theorem 1. *For any given hypothesis space Π with logging policy $\pi_0 \in \Pi$, there exists a reward distribution $\mathcal{P}_r$ with support in $[r_{min}, r_{max}]$ such that in the limit of infinite training data, ERM using IPS over the logged data $\mathcal{D} \sim P(\mathcal{X}) \times \pi_0(\cdot|\mathcal{X}) \times \mathcal{P}_r$ can select a policy $\hat{\pi} \in \arg\max_{\pi \in \Pi} \mathbb{E}_{\mathcal{D}}[\hat{R}_{IPS}(\pi)]$ that is at least*

$$(r_{max} - r_{min}) \max_{\pi \in \Pi} \mathcal{D}_{\mathcal{X}}(\pi|\pi_0)$$

suboptimal.

Proof. For any given hypothesis space Π and logging policy π_0, define a deterministic reward distribution $\mathcal{P}_r$ supported in $[r_{min}, r_{max}]$ as following: for all context x, $r(x, y) = \delta(x, y) = r_{min}$ for $y \in \mathcal{U}(x, \pi_0)^c$ and $r(x, y) = \delta(x, y) = r_{max}$ for $y \in \mathcal{U}(x, \pi_0)$. Let $\tilde{\pi} \in \arg\max_{\pi \in \Pi} \mathcal{D}_{\mathcal{X}}(\pi|\pi_0)$ and $\pi^* \in \arg\max_{\pi \in \Pi} R(\pi)$, then we have the following lower bound for $R(\pi^*)$:

$$\begin{aligned} R(\pi^*) &\geq R(\tilde{\pi}) \\ &= \mathop{\mathbb{E}}_{x} \left[\sum_{y \in \mathcal{U}(x, \pi_0)} r_{max} + \sum_{y \in \mathcal{U}(x, \pi_0)^c} r_{min} \right] \\ &= r_{max} \max_{\pi \in \Pi} \mathcal{D}_{\mathcal{X}}(\pi|\pi_0) + r_{min}(1 - \max_{\pi \in \Pi} \mathcal{D}_{\mathcal{X}}(\pi|\pi_0)) \end{aligned} \tag{6}$$

where the first inequality follows from the definition of π^*, the first and second equality is based on the specific reward distribution $\mathcal{P}_r$ and the definition of $\tilde{\pi}$.

In the following we will show that for any $\hat{\pi}$ learned by the expectation of ERM (or in the limit of infinite amount data), i.e., $\hat{\pi} \in \arg\max \mathbb{E}_{\mathcal{D}}[\hat{R}_{IPS}(\pi)]$, $\hat{\pi}$ have the same support as π_0.

$$\begin{aligned} \mathop{\mathbb{E}}_{\mathcal{D}}[\hat{R}_{IPS}(\pi)] &= \mathop{\mathbb{E}}_{x} \left[\sum_{y \in \mathcal{U}(x, \pi_0)^c} \pi(y|x) r_{min} \right] \\ &= r_{min} \mathop{\mathbb{E}}_{x} \left[\sum_{y \in \mathcal{U}(x, \pi_0)^c} \pi(y|x) \right] \leq r_{min} \end{aligned} \tag{7}$$

for all $\pi \in \Pi$, then it is easy to see $\pi_0 \in \Pi$ is one of the solution of ERM. Actually for any $\hat{\pi} \in \arg\max \mathbb{E}_{\mathcal{D}}[\hat{R}^{\mathcal{D}}_{IPS}(\pi)]$,

$$\mathop{\mathbb{E}}_{x} \left[\sum_{y \in \mathcal{U}(x, \pi_0)^c} \pi(y|x) \right] = 1$$

and it gives us that any solution of the ERM has exactly the same support as π_0, then we have $R(\hat{\pi}) = r_{min}$ for $\hat{\pi} \in \arg\max \mathbb{E}_{\mathcal{D}}[\hat{R}_{IPS}(\pi)]$.

Combining the lower bound for $R(\pi^*)$ and $R(\hat{\pi}) = r_{min}$, we have

$$\begin{aligned} R(\pi^*) - R(\hat{\pi}) &\geq r_{max} \max_{\pi \in \Pi} \mathcal{D}_{\mathcal{X}}(\pi|\pi_0) \\ &\quad + r_{min}(1 - \max_{\pi \in \Pi} \mathcal{D}_{\mathcal{X}}(\pi|\pi_0)) - r_{min} \\ &= (r_{max} - r_{min}) \max_{\pi \in \Pi} \mathcal{D}_{\mathcal{X}}(\pi|\pi_0) \end{aligned} \tag{8}$$

□

To illustrate the theorem, consider a problem with rewards $r \in [-1, 0]$. Furthermore, consider a policy space Π that contains a good policy π_g with $R(\pi_g) = -0.1$ and a bad policy π_b with $R(\pi_b) = -0.7$. If policy π_b has support divergence $\mathcal{D}_{\mathcal{X}}(\pi_b|\pi_0) = 0.6$ or larger, then ERM may return the bad π_b instead of π_g even with infinite amounts of training data.

Note that it is sufficient to merely have one policy in Π that has large support deficiency to achieve this suboptimality. It is therefore crucial to control the support divergence $\mathcal{D}_{\mathcal{X}}(\pi|\pi_0)$ uniformly over all $\pi \in \Pi$, or to account for the suboptimality it can induce. To this effect, we explore three approaches in the following.

3.1 Safe learning by restricting the action space

The first and arguably most direct approach to reducing $\mathcal{D}_{\mathcal{X}}(\pi|\pi_0)$ is to disallow any action that has zero support under the logging policy. For the remaining action set, the logging policy has full support by definition. This restriction of the action set can be achieved by transforming each policy $\pi \in \Pi$ into a new policy that sets the probability of the unsupported actions to zero.

$$\pi(y|x) \longrightarrow \bar{\pi}(y|x) := \frac{\pi(y|x)\, \mathbf{1}_{\{y \notin \mathcal{U}(x, \pi_0)\}}}{1 - \sum_{y' \in \mathcal{U}(x, \pi_0)} \pi(y'|x)} \tag{9}$$

This results in a new policy space $\bar{\Pi}$. All $\bar{\pi} \in \bar{\Pi}$ have support divergence of zero $\mathcal{D}_{\mathcal{X}}(\bar{\pi}|\pi_0) = 0$ and ERM via IPS is guaranteed to be unbiased.

While this transformation of the policy space from Π to $\bar{\Pi}$ is conceptually straightforward, it has two potential drawbacks. First, restricting the action space without any exceptions may overly constrain the policies in $\bar{\Pi}$. In particular, if the optimal action y^* for a specific context x does not have support under the logging policy, no $\bar{\pi} \in \bar{\Pi}$ can ever choose y^* even if there are many observations of similar y's on similar context x'. The second drawback is computational. For every context x during training and during testing, the system needs to evaluate the logging policy $\pi_0(y|x)$ to compute the transformation from π to $\bar{\pi}$. This can be prohibitively expensive especially at test time, where – after multiple rounds of off-policy learning with data from previously learned policies – we would need to evaluate the whole sequence of previous logging policies to execute the learned policy.

3.2 Safe learning through reward extrapolation

As illustrated above, support deficiency is a problem of blind spots where we lack information about the rewards of some actions in some contexts. Instead of disallowing the unsupported actions like in the previous section, an alternative is to extrapolate the observed rewards to fill in the blind spots. To this effect, we propose the following augmented IPS estimator that imputes an extrapolated reward $\hat{\delta}(x, y)$ for each unsupported action $y \in \mathcal{U}(x, \pi_0)$.

$$\hat{R}^{\hat{\delta}}_{IPS}(\pi) = \frac{1}{n} \sum_{i=1}^{n} \left[\frac{\pi(y_i|x_i)}{\pi_0(y_i|x_i)} r_i + \sum_{y \in \mathcal{U}(x_i, \pi_0)} \pi(y|x_i) \hat{\delta}(x_i, y) \right] \tag{10}$$

In general, $\hat{\delta}(x, y)$ can be any function that maps $\mathcal{X} \times \mathcal{Y}$ to $\mathbb{R}$. The higher the quality of $\hat{\delta}(x, y)$, the better the evaluation accuracy of the associated augmented IPS estimator. In the following proposition, we formally characterize the bias of the augmented IPS estimator for any given reward extrapolation $\hat{\delta}(x, y)$. We denote the mean of the reward r for context x and action y with $\delta(x, y) = \mathbb{E}_{r \sim P(r|x,y)}[r]$. Furthermore, let $\Delta(x, y) := \hat{\delta}(x, y) - \delta(x, y)$ denote the error of the reward extrapolation for each x and y.

Proposition 2. *Given contexts $x_1, x_2, \ldots, x_n$ drawn i.i.d from the unknown distribution $P(\mathcal{X})$, for action y_i drawn independently from logging policy π_0 with probability $\pi_0(y_i|x_i)$, the bias of the empirical risk defined in Equation (10) is $\mathbb{E}_x[\sum_{y \in \mathcal{U}(x, \pi_0)} \pi(y|x) \Delta(x, y)]$.*

The proof is provided in Appendix A.2. In this way we can learn in the original action and policy space, but mitigate the effect of the support deficiency by explicitly incorporating the extrapolated reward $\hat{\delta}(x, y)$. We explore two choices for $\hat{\delta}(x, y)$ in the following, which provide different types of guarantees.

Conservative Extrapolation. In practice, the logging policy is likely to put zero probability on actions that have low reward, since this minimizes the user impact of data collection. This means that precisely those bad actions are likely to not be supported in the logging policy. A key danger of blind spots regarding those actions is that naive IPS training will inadvertently learn a policy that selects those actions. This can be avoided by being maximally conservative about unsupported actions and imputing the lowest possible reward [19].

$$\forall x \, \forall y \in \mathcal{U}(x, \pi_0) : \hat{\delta}(x, y) = r_{min}$$

Intuitively, by imposing the worst possible reward for the unsupported actions, the learning algorithm will aim to avoid these low-reward areas. However, unlike for the $\bar{\pi}$ policies resulting from the restricted action space, the learned policy is not strictly prohibited from choosing unsupported actions – it is merely made aware of the maximum loss that the action may incur. Note that for problems where $r_{min} = 0$, the naive IPS estimator is identical to conservative extrapolation since the second term in Equation (10) is zero.

Regression Extrapolation. Instead of extrapolating with the worst-case reward, we may have additional prior knowledge in the form of a model-based estimate that reduces the bias. In particular, we explore using a regression estimate

$$\hat{\delta} = \arg\min_{\hat{\delta}^\theta} \frac{1}{n} \sum_{i=1}^{n} (\hat{\delta}^\theta(x_i, y_i) - r_i)^2$$

that extrapolates from the observed data $\mathcal{D}$. Typically, $\hat{\delta}^\theta$ comes from a parameterized class of regression functions (e.g. linear, deep networks).[1] Other regression objectives could also be used, such as weighted linear regression that itself uses importance sampling as weights [6]. But, fundamentally, all regression approaches assume

[1] In our experiments, we use deep neural networks with details shown in Appendix B.

Algorithm 1: Data Augmentation

input: original logged dataset $\mathcal{D}$, replaycount k, reward estimate $\hat{\delta}(x, y)$;
output: additional augmented dataset $\mathcal{D}'$;
initialization: $\mathcal{D}' = \emptyset$;
for $j = 1, \ldots, k$ **do**
  for $i = 1, \ldots, n$ **do**
    Define U_{x_i} to be the uniform distribution over $\mathcal{U}(x_i, \pi_0)$;
    Draw $y \sim U_{x_i}$;
    $\mathcal{D}' = \mathcal{D}' \bigcup \{x_i, y, \hat{\delta}(x_i, y), \frac{1}{|\mathcal{U}(x_i, \pi_0)|}\}$;
  end
end

that the regression model is not misspecified and that it can thus extrapolate well.

Note that the IPS part of Equation (10) can be exchanged for other estimators. In particular, we note that doubly robust (DR) [5] naturally performs a form of regression extrapolation. As the following decomposition shows, DR imputes the extrapolated reward $\hat{\delta}(x_i, y)$ for the unsupported actions $y \in \mathcal{U}(x_i, \pi_0)$.

$$\begin{aligned}
\hat{R}^{\hat{\delta}}_{DR} &= \frac{1}{n}\sum_{i=1}^{n}\left[\sum_{y\in\mathcal{Y}} \pi(y|x_i)\hat{\delta}(x_i, y) + \frac{\pi(y_i|x_i)}{\pi_0(y_i|x_i)}(r_i - \hat{\delta}(x_i, y_i))\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\left[\sum_{y\in\mathcal{U}(x_i,\pi_0)^c} \pi(y|x_i)\hat{\delta}(x_i, y) + \frac{\pi(y_i|x_i)}{\pi_0(y_i|x_i)}(r_i - \hat{\delta}(x_i, y_i))\right. \\
&\quad \left. + \sum_{y\in\mathcal{U}(x_i,\pi_0)} \pi(y|x_i)\hat{\delta}(x_i, y)\right] \qquad (11)
\end{aligned}$$

A similar decomposition also exists for the CAB [22] estimator, showing that both DR and CAB belong to the class of Regression Extrapolation estimators.

Efficient Approximation. Evaluating the augmented IPS estimator from Equation (10) can be computationally expensive if the number of unsupported actions in $\mathcal{U}(x, \pi_0)$ is large. To overcome this problem, we propose to use sampling to estimate the expected reward on the unsupported actions, which can be thought of as augmenting the dataset $\mathcal{D}$ with additional observations where the logging policy has zero support. In particular, we propose the data-augmentation procedure detailed in Algorithm 1. With the additional bandit data $\mathcal{D}' = \{x'_j, y'_j, \hat{\delta}(x'_j, y'_j), p'_j\}_{j=1}^{m}$ from Algorithm 1, the new objective is [2]

$$\arg\min_{\pi\in\Pi}\left\{\frac{1}{n}\sum_{i=1}^{n}\frac{\pi(y_i|x_i)}{\pi_0(y_i|x_i)}r_i + \frac{1}{m}\sum_{j=1}^{m}\frac{\pi(y'_j|x'_j)}{p'_j}\hat{\delta}(x'_j, y'_j)\right\} \qquad (12)$$

In Appendix A.3, we show that the empirical risk in Equation (12) has the same expected value (over randomness in $\mathcal{D}$ and $\mathcal{D}'$) as $\hat{R}^{\hat{\delta}}_{IPS}(\mathcal{D})$ and can thus serve as an approximation for Equation (10).

[2]For the rest of the paper, we will use maximizing reward or minimizing loss interchangeably.

3.3 Safe Learning by Restricting the Policy Space

As motivated by Theorem 1, the risk of learning from support deficient data scales with the maximum support divergence $\mathcal{D}_{\mathcal{X}}(\pi|\pi_0)$ among the policies in the policy space Π. Therefore, our third approach restricts the policy space to the subset $\Pi^{\kappa} \subset \Pi$ that contains the policies $\pi \in \Pi$ with an acceptably low support divergence $\mathcal{D}_{\mathcal{X}}(\pi|\pi_0) \leq \kappa$.

$$\Pi^{\kappa} = \{\pi \in \Pi | \mathcal{D}_{\mathcal{X}}(\pi|\pi_0) \leq \kappa\} \qquad (13)$$

The parameter κ has an intuitive meaning. It specifies the maximum probability mass that a learned policy can place on unsupported actions. Typically the choice of κ is application dependent, limiting the maximum bias of the ERM procedure according to Proposition 2 while not explicitly torquing the rewards like in conservative reward imputation. A key challenge, however, is implementing this restriction of the hypothesis space, such that the ERM learner

$$\hat{\pi} = \arg\max_{\pi\in\Pi^{\kappa}}[\hat{R}_{IPS}(\pi)]$$

only considers the subset $\Pi^{\kappa} \subset \Pi$.

In particular, we do not have access to the context distribution $P(\mathcal{X})$ for calculating $\mathcal{D}_{\mathcal{X}}(\pi|\pi_0)$, nor would it be possible to enumerate all $\pi \in \Pi$ to check the condition $\mathcal{D}_{\mathcal{X}}(\pi|\pi_0) \leq \kappa$, which itself requires a possibly infeasible iteration over all actions. The following theorem gives us an efficient way of estimating and controlling $\mathcal{D}_{\mathcal{X}}(\pi|\pi_0)$ without explicit knowledge of $P(\mathcal{X})$ or access to the logging policy π_0 beyond the logged propensities.

Theorem 2. *For contexts x_i drawn i.i.d from $P(\mathcal{X})$, action y_i drawn from logging policy $\pi_0(\mathcal{Y}|x_i)$, we define $S_{\mathcal{D}}(\pi|\pi_0) = \frac{1}{n}\sum_{i=1}^{n}\frac{\pi(y_i|x_i)}{\pi_0(y_i|x_i)}$. For any policy π it holds that*

$$\mathop{\mathbb{E}}_{x\sim P(\mathcal{X})}\mathop{\mathbb{E}}_{y\sim\pi_0(\mathcal{Y}|x)}[S_{\mathcal{D}}(\pi|\pi_0)] + \mathcal{D}_{\mathcal{X}}(\pi|\pi_0) = 1 \qquad (14)$$

The proof is shown in Appendix A.4. Using this theorem, the following proposition gives us an efficient way of implementing the constraint $\mathcal{D}_{\mathcal{X}}(\pi|\pi_0) \leq \kappa$ via $1 - S_{\mathcal{D}}(\pi|\pi_0)$.

Proposition 3. *For any given $\kappa \in (0, 1)$, and for ϵ with $0 < \epsilon < \kappa/2$, let p_{min} denote the minimum propensity on the supported set with $p_{min} = min_{x,y\in\mathcal{U}(x,\pi_0)^c}\pi_0(y|x)$, then with probability larger than $1 - 2\exp(-2n\epsilon^2 p^2_{min})$, the constraint $1 - \kappa + \epsilon \leq S_{\mathcal{D}}(\pi|\pi_0) \leq 1 - \epsilon$ will ensure $0 \leq \mathcal{D}_{\mathcal{X}}(\pi|\pi_0) \leq \kappa$.*

The proof is provided in Appendix A.5. We can thus use $1 - S_{\mathcal{D}}(\pi|\pi_0)$ as a surrogate for $\mathcal{D}_{\mathcal{X}}(\pi|\pi_0)$ in the IPS training objective (or similar objectives like DR, clipped IPS, or CAB).

$$\begin{aligned}
&\arg\min_{\pi_w\in\Pi}\frac{1}{n}\sum_{i=1}^{n}\frac{\pi_w(y_i|x_i)}{\pi_0(y_i|x_i)}r_i \\
&\text{subject to } 1 - \kappa + \epsilon \leq \frac{1}{n}\sum_{i=1}^{n}\frac{\pi_w(y_i|x_i)}{\pi_0(y_i|x_i)} \leq 1 - \epsilon
\end{aligned} \qquad (15)$$

Using Lagrange multipliers, an equivalent dual form of Equation (15) is:

$$\max_{u_1,u_2\geq 0}\min_{\pi_w\in\Pi}\frac{1}{n}\sum_{i=1}^{n}\frac{\pi_w(y_i|x_i)}{\pi_0(y_i|x_i)}(r_i + u_1 - u_2) - u_1(1-\epsilon) + u_2(1-\kappa+\epsilon) \qquad (16)$$

For each fixed (u_1, u_2) pair, the inner minimization objective is ERM with IPS where the reward is shifted by $k = (u_1 - u_2)$. So, we can select k, solve (16), and compute the corresponding κ afterwards. To achieve a desired κ, we can use any suitable root finding method for k. We simply perform a grid search over $k = u_1 - u_2$.

Empirical Model Selection for k. While we may have a desired risk tolerance κ for selecting k in some applications, on others we may want to select the k that maximize performance on a validation set. This again requires some estimate of the reward on the unsupported actions. We thus explore Conservative Extrapolation, DM, and the following model-independent approach we call *MinSup*. MinSup aims to get the best model-free estimate of the reward on the unsupported-action set that the available data admits. In particular, we construct a minimally supported policy π_{MinSup} that is closest to a policy that only takes unsupported actions while still having full support. The construction is as follows: for each context x, we greedily put all the probability p on the action that has the lowest propensity, while keeping the IPS weights $(\frac{p}{\pi_0(y|x_i)})$ to be bounded by 100. If $p \leq 1$, we continue this procedure with the next-lowest propensity until we have distributed all of the probability mass. We then estimate the value of the constructed policy π_{MinSup} using IPS to arrive at the following estimator for any target policy π, which substitutes $\hat{R}_{IPS}(\pi_{MinSup})$ for the missing support of $\hat{R}_{IPS}(\pi)$.

$$\hat{R}_{MinSup}(\pi) = \hat{R}_{IPS}(\pi) + \Big(1 - S_{\mathcal{D}}(\pi|\pi_0)\Big)\hat{R}_{IPS}(\pi_{MinSup}) \quad (17)$$

Note that IPS is unbiased for π_{MinSup} since it has the same support set as π_0. Furthermore, it has bounded variance since the IPS weights are bounded by construction. For all the empirical evaluations in this paper, we use MinSup to select the optimal k. We also compare MinSup to DM and Conservative Extrapolation for this model-selection problem in Section 4.1.

Practical Considerations. Among the methods we proposed for dealing with support deficiency, the Policy Restriction approach is easy to implement, does not require an additional regression model with unknown bias, and it does not require access to the logging policy during training or testing. In particular, the form of the inner objective coincides with that of BanditNet [11], which is known to work well for deep network training by controlling propensity overfitting [24].

4 EMPIRICAL EVALUATION

We empirically analyze and compare the effectiveness and robustness of the three approaches: restricting the action space, reward extrapolation, and restricting the policy space. We use two real-world datasets, namely the image-classification dataset CIFAR10 [13] and the credit-card fraud dataset of [4], from which we generate various degrees of support deficient bandit data.

The experiments are set up as follows. We first create a train-validation-test split for both datasets. The training set is used to generate bandit datasets for learning, the validation set is used to generate bandit datasets for model selection, and the full-information test set serves as ground truth for evaluating the learned policies. To simulate bandit feedback for the CIFAR10 dataset, our experiment setup follows traditional supervised → bandit conversion for multi-class classification datasets [2]. To not limit our evaluation to binary multi-class rewards, we choose a different methodology for the credit-card dataset by designating some features as corresponding to actions and rewards. More details are given in Appendix B.

For both logging and target policies, we train softmax policies (Equation (4)) where $f_w(x, y)$ is a neural network. We use the ResNet20 architecture [9] for CIFAR10, and a fully connected 2-layer network for the credit-card dataset. We then introduce a temperature parameter τ into the learned policy via $\tau f_w(x, y)$ to be able to control its stochasticity and support deficiency. In particular, we enforce zero support for some actions by clipping the propensities to 0 if they are below a threshold of $\epsilon = 0.01$. The larger τ, the higher the support deficiency. Note that setting the threshold at $\epsilon = 0.01$ allows us to control support without having to worry about variance control. More details are given in Appendix B.

The estimators that we examined in our experiments are: first, traditional baselines including naive IPS (Eq. (3)) and DM; second, the Action Restriction approach (Eq. (9)); third, three Reward Extrapolation methods (Conservative Extrapolation, Regression Extrapolation, and DR (Eq. (11)); as well as, fourth, the Policy Restriction approach (Eq. (16)). If not mentioned otherwise, MinSup is used to select the k parameter and all experiment results are averaged over 5 runs.

4.1 Experiments and Findings

The following experiments investigate the key properties of the estimators, and they inform our recommendations and conclusions.

How do the methods perform at different level of support deficiency? Figure 1 shows the test accuracy and support divergence $\mathcal{D}_\chi(\pi|\pi_0)$ as support deficiency increases. First, as expected, learning using naive IPS degrades on both datasets. Note that naive IPS coincides with Conservative Extrapolation in the left two columns, since both datasets are scaled to have a minimum reward of zero. In the rightmost column, however, we translated the rewards from $[0, 1]$ to $[-1, 0]$. This has a strong detrimental effect on naive IPS. IPS is inherently imputing reward 0, and the performance of naive IPS highly depends on the position of 0 in the range of the reward. Second, the Action Restriction approach also performs poorly. While its support divergence $\mathcal{D}_\chi(\pi|\pi_0)$ is zero and thus bias is not the problem, we conjecture that the best actions are often pruned from the action-restricted policy space $\bar{\Pi}$. Third, Regression Extrapolation tends to perform better than Conservative Extrapolation in our experiments. On both datasets, the DM model turns out to be quite good, which also benefits DR. However, on the credit-card dataset the regression seems better at ranking than at predicting the true reward, which explains why DM performs better than Regression Extrapolation. Fourth, the methods that performs well on both datasets are Policy Restriction and DR. Unlike all the other IPS-based methods, Policy Restriction performs well even under the translated rewards in the third column of Figure 1. This is because the objective of Policy Restriction coincides with that of BanditNet [11], which is known to remedy propensity overfitting due to the lack of equivariance of the IPS estimator [25].

How does regression-model misspecification affect the estimators? While DR performed well in the previous experiments, it has no

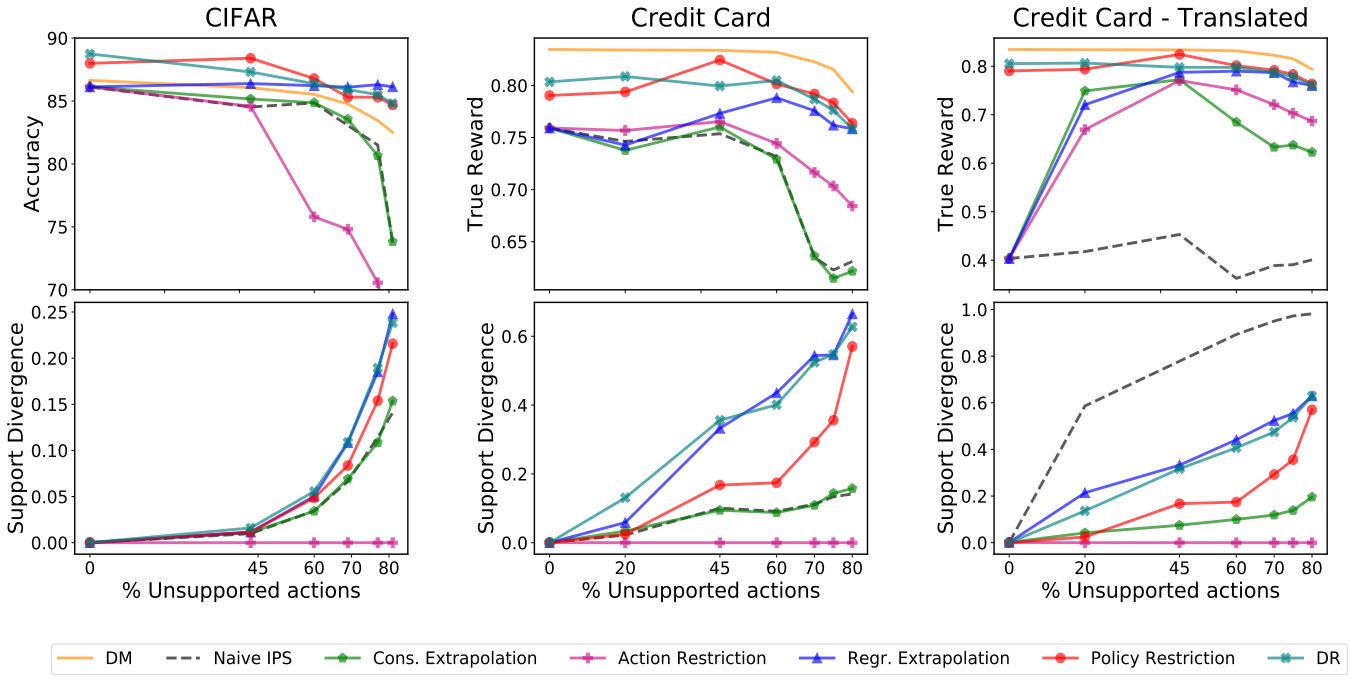

Figure 1: Test set accuracy and support divergence of the learned policies as we pick logging policies that have increasing fractions of unsupported actions.

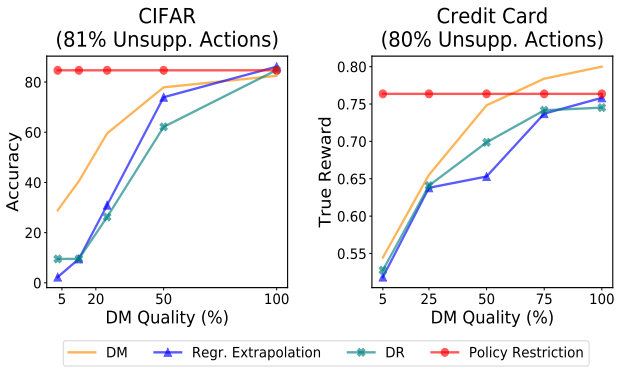

Figure 2: Test set accuracy for different levels of misspecification of the regression model.

mechanism for guarding against bias from model misspecification under support deficiency. The same limitation also applies to the other estimators that rely on regression imputation for the unsupported actions. We thus examine how different estimators deal with model misspecification. To simulate increasing levels of model misspecification, we train the regression function on subsets of input features of varying size. Results are shown in Figure 2. As the number of input feature decreases and misspecification thus increases, the effectiveness of DR, Regression Extrapolation and DM decreases substantially. Since Policy Restriction does not rely on any regression model, it is unaffected.

% Unsupp.	Oracle	DM	Cons. Extra.	MinSup
43	88.397	87.526	**88.397**	**88.397**
60	86.782	**86.782**	**86.782**	**86.782**
69	86.462	85.308	85.308	85.308
77	85.295	85.282	84.090	**85.295**
81	85.192	**85.192**	83.526	84.680

Table 1: Test error rates on the CIFAR10 data using the respective model selection method under varying levels of support deficiency in the logging policy.

How does the learning performance change with the amount of training data? Figure 3 shows how accuracy on the test set changes with the amount for training data for two levels of support deficiency. Policy Restriction is at least competitive with Regression Extrapolation, DR, and DM over most of the range. Action Restriction can take the least advantage of more data. This is plausible, since its maximum performance is limited by the available actions. For similar reasons, Conservative Extrapolation, and equivalently IPS, also flatten out, since they also tightly restrict the action space by imputing the minimum reward.

How effective is model selection for k? If there is no pre-specified risk tolerance τ that is derived from application requirements, Section 3.3 proposed to select the parameter k on a validation set. Table 1 shows how the proposed MinSup model selection criterion compares against model selection via DM and Conservative Extrapolation. We also report the skyline performance of an Oracle model selector which has access to the full-information validation

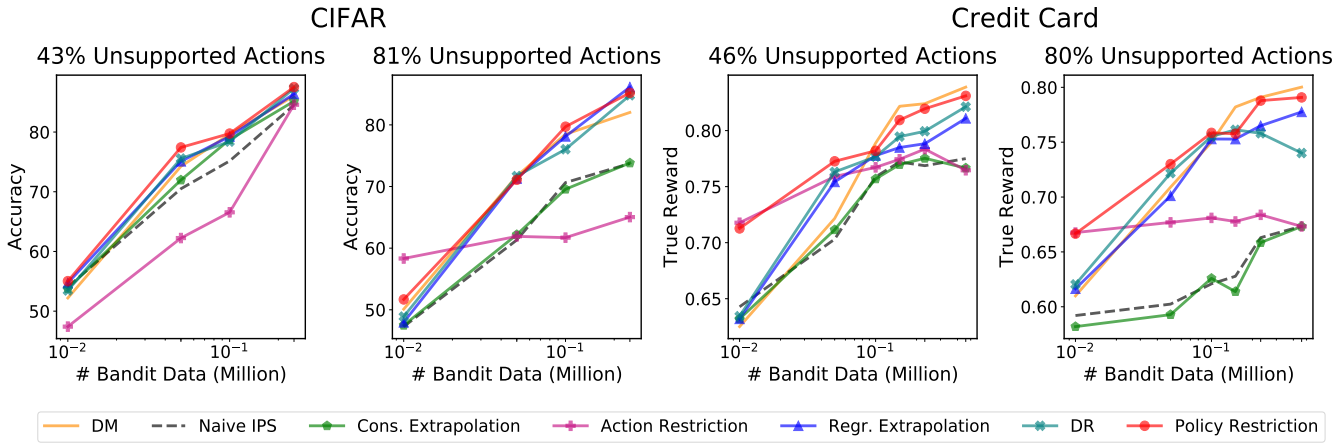

Figure 3: Test set accuracy as the amount of training data increases for two levels of support deficiency.

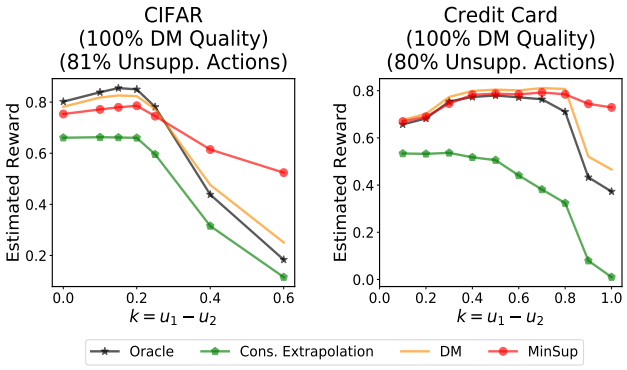

Figure 4: Comparison of estimators of validation set accuracy for the policies learned by Policy Restriction when the parameter k is varied.

set. Conservative extrapolation consistently underperforms in high-deficiency settings, while both MinSup and DM perform well as model selection criteria. This is explained by the plots in Figure 4, which show the value of the validation set estimates. While MinSup and DM have their optimum close to that of Oracle, the Conservative Extrapolation curve is substantially off. Additional results in Appendix B, however, show that model misspecification can again substantially affect the performance of model selection via DM.

How accurate is the estimator of support divergence? Policy restriction relies on the estimated support divergence, and we now evaluate the effectiveness of this estimator. The target policy is the uniform policy while the logging policies are varying in their support deficiency. We investigate the behaviour of $S_{\mathcal{D}}(\pi|\pi_0) + \mathcal{D}_{\mathcal{X}}(\pi|\pi_0)$ for increasing amounts of training data and different support deficiency of the corresponding logging policy. Results are shown in Figure 5 averaged over 10 runs. The figure shows that the sum converges to 1 and the variance of the estimate decreases as the amount of training data increases. The curves for different levels of support deficiency converge in a similar fashion and we conjecture it is due to the effect of clipping the propensity at the same threshold $\epsilon = 0.01$, which results in $p_{min} = 0.01$ in the bound shown in Proposition 3.

5 DISCUSSION AND RECOMMENDATIONS

We now discuss the advantages and disadvantages of the various approaches to dealing with support deficiency, and provide guidance for their use in practice.

While Action Restriction may be the the most natural and direct approach at first glance, we find that it has substantial drawbacks. In particular, it is computationally expensive since we need to calculate the propensity under the logging policy $\pi_0(y|x)$ for every x not only at training time, but also at testing time. This is particularly problematic when policies are updated in a frequent manner, since we need to revisit the whole sequence of past executed logging policies at test time. Furthermore, its learning performance is substantially worse than other methods, since actions are limited to an overly conservative regime that enforces zero support divergence.

Conservative Extrapolation allows the target policy to select actions that have zero support under the logging policy, and performance tends to be better than for Action Restriction. However, Conservative Extrapolation typically performs worse than Policy Restriction, Regression Extrapolation, and DR. All methods are more efficient that Action Restriction at test time as they do not require evaluating the old logging policy. During training, however, all Reward Imputation methods (i.e. Conservative Extrapolation, Regression Extrapolation, and DR) need to evaluate all actions, which can be expensive but ameliorated through sampling. A key risk of both Regression Extrapolation and DR is that they rely on a regression model, which can introduce biases from model misspecification that are fundamentally unknown. The estimators provide no mechanism for guarding against such biases.

Policy Restriction does not rely on a regression model, which eliminates the need for training such a model. Furthermore, it is the only method that allows flexible risk control through the parameter κ or k respectively. If the application does not provide a risk threshold, k can be selected empirically via MinSup. From a computational

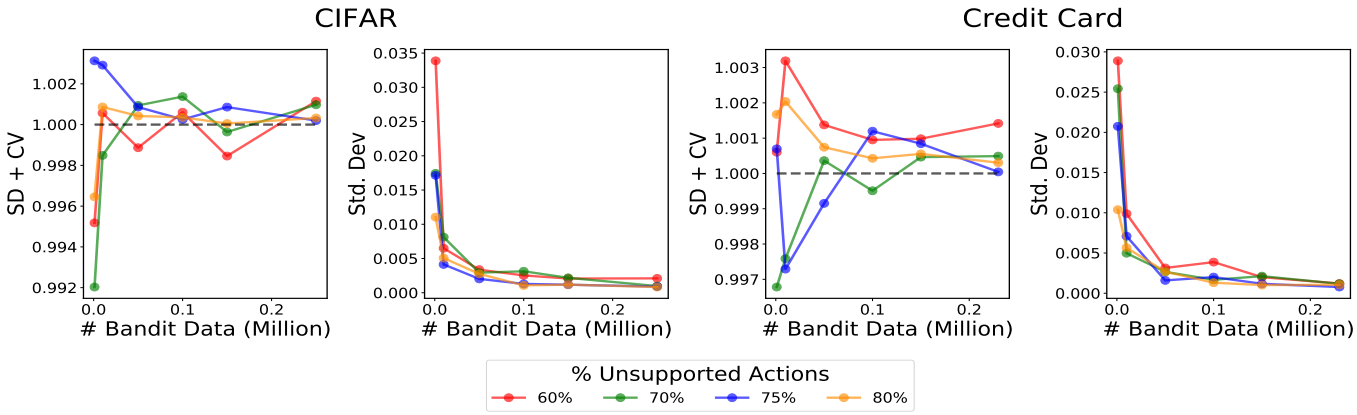

Figure 5: Estimate of $S_{\mathcal{D}}(\pi|\pi_0) + \mathcal{D}_{\mathcal{X}}(\pi|\pi_0)$ and its estimated standard deviation as the amount of training data increases.

perspective, Policy Restriction is efficient at both training and test time, and it has minimal logging requirements as it only requires the logged propensity of the chosen action. Since it consistently showed at least competitive generalization performance across a wide range of settings, we conclude that it is a preferable choice for practical applications – especially when there are concerns about model misspecification.

6 CONCLUSIONS

This paper presented the first comprehensive analysis of support deficiency in off-policy learning for contextual bandits. In particular, it identified and explored three approaches to dealing with support deficiency: restricting the action space, reward extrapolation, and restricting the policy space. The paper characterized the theoretical properties of these approaches, and empirically evaluated their performance and robustness. We conclude that restricting the policy space is particularly effective, since it provides explicit risk control, performs well in terms of learning performance, and it is easy and efficient to implement.

ACKNOWLEDGMENTS

This research was supported in part by NSF Award IIS-1901168 and by a Bloomberg Fellowship. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. 2014. Taming the monster: A fast and simple algorithm for contextual bandits. In *ICML*.
[2] Alina Beygelzimer and John Langford. 2009. The offset tree for learning with partial labels. In *KDD*. ACM, 129–138.
[3] Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. 2013. Counterfactual reasoning and learning systems: The example of computational advertising. *JMLR* 14, 1 (2013), 3207–3260.
[4] Andrea Dal Pozzolo, Olivier Caelen, Reid A Johnson, and Gianluca Bontempi. 2015. Calibrating probability with undersampling for unbalanced classification. In *2015 SSCI*. IEEE, 159–166.
[5] Miroslav Dudík, John Langford, and Lihong Li. 2011. Doubly Robust Policy Evaluation and Learning. In *ICML*.
[6] Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. 2018. More Robust Doubly Robust Off-policy Evaluation. In *ICML*. 1446–1455.
[7] Scott Fujimoto, David Meger, and Doina Precup. 2018. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900* (2018).
[8] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. 2004. Variance reduction techniques for gradient estimates in reinforcement learning. *JMLR* 5, Nov (2004), 1471–1530.
[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.
[10] Nan Jiang and Lihong Li. 2016. Doubly Robust Off-policy Value Evaluation for Reinforcement Learning. In *ICML*. 652–661.
[11] T. Joachims, A. Swaminathan, and M. de Rijke. 2018. Deep Learning with Logged Bandit Feedback. In *ICLR*.
[12] T. Joachims, A. Swaminathan, and T. Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In *WSDM*.
[13] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2014. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html* 55 (2014).
[14] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. In *NeurIPS*. 11761–11771.
[15] John Langford and Tong Zhang. 2008. The epoch-greedy algorithm for multi-armed bandits with side information. In *NeurIPS*. 817–824.
[16] Romain Laroche, Paul Trichelair, and Rémi Tachet des Combes. 2017. Safe policy improvement with baseline bootstrapping. *arXiv preprint arXiv:1712.06924* (2017).
[17] Lihong Li, Shunbao Chen, Jim Kleban, and Ankur Gupta. 2015. Counterfactual estimation and optimization of click metrics in search engines: A case study. In *WWW*. ACM, 929–934.
[18] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. 2011. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM*. ACM, 297–306.
[19] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. 2019. Off-Policy Policy Gradient with State Distribution Correction. *arXiv preprint arXiv:1904.08473* (2019).
[20] Ben London and Ted Sandler. 2019. Bayesian Counterfactual Risk Minimization. In *ICML*. 4125–4133.
[21] Alex Strehl, John Langford, Lihong Li, and Sham M Kakade. 2011. Learning from Logged Implicit Exploration Data. In *NeurIPS*.
[22] Yi Su, Lequn Wang, Michele Santacatterina, and Thorsten Joachims. 2019. CAB: Continuous Adaptive Blending for Policy Evaluation and Learning. In *ICML*. 6005–6014.
[23] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.
[24] A. Swaminathan and T. Joachims. 2015. Batch Learning from Logged Bandit Feedback through Counterfactual Risk Minimization. *JMLR* 16 (Sep 2015), 1731–1755. Special Issue in Memory of Alexey Chervonenkis.
[25] A. Swaminathan and T. Joachims. 2015. The Self-Normalized Estimator for Counterfactual Learning. In *NeurIPS*.
[26] Philip Thomas and Emma Brunskill. 2016. Data-efficient Off-policy Policy Evaluation for Reinforcement Learning. In *ICML*.
[27] Yu-Xiang Wang, Alekh Agarwal, and Miroslav Dudik. 2017. Optimal and Adaptive Off-policy Evaluation in Contextual Bandits. In *ICML*.
[28] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.
[29] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

A APPENDIX: PROOFS

A.1 Proof of Proposition 1

Proposition 1. *Given contexts $x \sim P(\mathcal{X})$ and logging policy $\pi_0(\mathcal{Y}|x)$, the bias of $\hat{R}_{IPS}$ for target policy $\pi(\mathcal{Y}|x)$ is equal to the expected reward on the unsupported action sets, i.e.,*

$$bias(\hat{R}_{IPS}(\pi)) = \mathop{\mathbb{E}}_{x}\Big[- \sum_{y \in \mathcal{U}(x,\pi_0)} \pi(y|x)\delta(x,y)\Big].$$

Proof. Recall $\delta(x,y) = \mathbb{E}_r[r(x,y)|x,y]$, and logged data $\mathcal{D} \sim \mathcal{P}_\mathcal{X} \times \pi_0(\cdot|\mathcal{X}) \times \mathcal{P}_r$.

$$\begin{aligned} bias(\hat{R}_{IPS}(\pi)) &= \mathop{\mathbb{E}}_{\mathcal{D}}\big[\hat{R}_{IPS}(\pi)\big] - R(\pi) \\ &= \mathop{\mathbb{E}}_{x}\Bigg[\sum_{y \in (\mathcal{U}(x,\pi_0))^c} \pi_0(y|x)\frac{\pi(y|x)}{\pi_0(y|x)}\delta(x,y) \\ &\quad - \sum_{y \in \mathcal{Y}} \pi(y|x)\delta(x,y)\Bigg] \\ &= \mathop{\mathbb{E}}_{x}\Big[- \sum_{y \in \mathcal{U}(x,\pi_0)} \pi(y|x)\delta(x,y)\Big] \end{aligned} \tag{18}$$

□

A.2 Proof of Proposition 2

Proposition 2. *Given contexts $x_1, x_2, \ldots, x_n$ drawn i.i.d from the unknown distribution $P(\mathcal{X})$, for action y_i drawn independently from logging policy π_0 with probability $\pi_0(y_i|x_i)$, the bias of the empirical risk defined in Equation (10) is $\mathbb{E}_x[\sum_{y \in \mathcal{U}(x,\pi_0)} \pi(y|x)\Delta(x,y)]$.*

Proof. Based on the definition of $\hat{R}^{\hat{\delta}}_{IPS}(\pi)$:

$$\begin{aligned} &\mathop{\mathbb{E}}_{\mathcal{D}}[\hat{R}^{\hat{\delta}}_{IPS}(\pi)] - R(\pi) \\ &= \mathop{\mathbb{E}}_{x}\Bigg[\sum_{y \in \mathcal{U}(x,\pi_0)^c} \pi(y|x)\delta(x,y) + \sum_{y \in \mathcal{U}(x,\pi_0)} \pi(y|x)\hat{\delta}(x,y)\Bigg] - R(\pi) \\ &= \mathop{\mathbb{E}}_{x}\Bigg[\sum_{y \in \mathcal{U}(x,\pi_0)} \pi(y|x)(\hat{\delta}(x,y) - \delta(x,y))\Bigg] \\ &= \mathop{\mathbb{E}}_{x}\Bigg[\sum_{y \in \mathcal{U}(x,\pi_0)} \pi(y|x)\Delta(x,y)\Bigg] \end{aligned} \tag{19}$$

The second equality is based on the decomposition of $R(\pi)$, and the last one is based on the definition of $\Delta(x,y) := \hat{\delta}(x,y) - \delta(x,y)$ for all $x \in \mathcal{X}, y \in \mathcal{Y}$. □

A.3 Proof of Efficient Approximation

Claim 1. *The empirical risk defined by in Equation (12) has the same expectation (over randomness in $\mathcal{D}$ and sampling) as $\hat{R}^{\hat{\delta}}_{IPS}(\mathcal{D})$.*

Proof. Taking the expectation of empirical risk defined in Equation (12):

$$\begin{aligned} &\mathbb{E}\Bigg[\frac{1}{n}\sum_{i=1}^{n}\frac{\pi(y_i|x_i)}{\pi_0(y_i|x_i)}r_i + \frac{1}{m}\sum_{j=1}^{m}\frac{\pi(y_j|x_j)}{p_j}\hat{\delta}(x_j,y_j)\Bigg] \\ &= \mathop{\mathbb{E}}_{x}\Bigg[\sum_{y \in \mathcal{U}(x,\pi_0)^c} \pi_0(y|x)\frac{\pi(y|x)}{\pi_0(y|x)}\delta(x,y)\Bigg] \\ &+ \mathop{\mathbb{E}}_{x}\Bigg[\sum_{y \in \mathcal{U}(x,\pi_0)} \frac{1}{|\mathcal{U}(x,\pi_0)|}\frac{\pi(y|x)}{\frac{1}{|\mathcal{U}(x,\pi_0)|}}\hat{\delta}(x,y)\Bigg] \\ &= \mathop{\mathbb{E}}_{x}\Bigg[\sum_{y \in \mathcal{U}(x,\pi_0)^c} \pi(y|x)\delta(x,y)\big] + \mathop{\mathbb{E}}_{x}\big[\sum_{y \in \mathcal{U}(x,\pi_0)} \pi(y|x)\hat{\delta}(x,y)\Bigg] \end{aligned} \tag{20}$$

Now we will show it has the same expectation with $\hat{R}^{\hat{\delta}}_{IPS}(\pi)$

$$\begin{aligned} &\mathop{\mathbb{E}}_{\mathcal{D}}\Bigg[\frac{\pi(y_i|x_i)}{\pi_0(y_i|x_i)}r_i + \sum_{y \in \mathcal{U}(x_i,\pi_0)} \pi(y|x_i)\hat{\delta}(x_i,y)\Bigg] \\ &= \mathop{\mathbb{E}}_{x}\Bigg[\mathop{\mathbb{E}}_{y\sim\pi_0}\Big[\frac{\pi(y|x)}{\pi_0(y|x)}\delta(x,y)\Big] + \sum_{y' \in \mathcal{U}(x,\pi_0)} \pi(y'|x)\hat{\delta}(x,y')\Bigg] \\ &= \mathop{\mathbb{E}}_{x}\Bigg[\sum_{y \in \mathcal{U}(x,\pi_0)^c} \pi_0(y|x)\frac{\pi(y|x)}{\pi_0(y|x)}\delta(x,y) + \sum_{y' \in \mathcal{U}(x,\pi_0)} \pi(y'|x)\hat{\delta}(x,y')\Bigg] \\ &= \mathop{\mathbb{E}}_{x}\Bigg[\sum_{y \in \mathcal{U}(x,\pi_0)^c} \pi(y|x)\delta(x,y)\Bigg] + \mathop{\mathbb{E}}_{x}\Bigg[\sum_{y \in \mathcal{U}(x,\pi_0)} \pi(y|x)\hat{\delta}(x,y)\Bigg] \end{aligned} \tag{21}$$

The proof is done by comparing Equation (A.3) and Equation (21). □

A.4 Proof of Theorem 2

Theorem 2. *For contexts x_i drawn i.i.d from $P(\mathcal{X})$, action y_i drawn from logging policy $\pi_0(\mathcal{Y}|x_i)$, we define $S_\mathcal{D}(\pi|\pi_0) = \frac{1}{n}\sum_{i=1}^{n}\frac{\pi(y_i|x_i)}{\pi_0(y_i|x_i)}$. For any policy π it holds that*

$$\mathop{\mathbb{E}}_{x\sim P(\mathcal{X})}\mathop{\mathbb{E}}_{y\sim\pi_0(\mathcal{Y}|x)}[S_\mathcal{D}(\pi|\pi_0)] + \mathcal{D}_\mathcal{X}(\pi|\pi_0) = 1 \tag{14}$$

Proof.

$$\begin{aligned} &\mathop{\mathbb{E}}_{x\sim P(\mathcal{X})}\mathop{\mathbb{E}}_{y\sim\pi_0(\cdot|x)}[S_\mathcal{D}(\pi|\pi_0)] + \mathcal{D}_\mathcal{X}(\pi|\pi_0) \\ &= \mathop{\mathbb{E}}_{x}\Bigg[\sum_{y \in \mathcal{U}(x,\pi_0)^c} \pi_0(y|x)\frac{\pi(y|x)}{\pi_0(y|x)}\Bigg] + \mathcal{D}_\mathcal{X}(\pi|\pi_0) \\ &= \mathop{\mathbb{E}}_{x}\Bigg[\sum_{y \in \mathcal{U}(x,\pi_0)^c} \pi(y|x)] + \mathop{\mathbb{E}}_{x}[\sum_{y \in \mathcal{U}(x,\pi_0)} \pi(y|x)\Bigg] \\ &= \mathop{\mathbb{E}}_{x}\Bigg[\sum_{y \in \mathcal{Y}} \pi(y|x)\Bigg] = 1 \end{aligned} \tag{22}$$

The first equality is based on definition of $S_\mathcal{D}(\pi|\pi_0)$ and the second equality is based on definition of support divergence. □

A.5 Proof of Proposition 3

Proposition 3. *For any given $\kappa \in (0,1)$, and for ϵ with $0 < \epsilon < \kappa/2$, let p_{min} denote the minimum propensity on the supported set with $p_{min} = min_{x,y\in\mathcal{U}(x,\pi_0)^c}\pi_0(y|x)$, then with probability larger than $1 - 2\exp(-2n\epsilon^2 p_{min}^2)$, the constraint $1 - \kappa + \epsilon \le S_\mathcal{D}(\pi|\pi_0) \le 1 - \epsilon$ will ensure $0 \le \mathcal{D}_\mathcal{X}(\pi|\pi_0) \le \kappa$.*

Proof. Recall $S_{\mathcal{D}}(\pi|\pi_0) = \frac{1}{n}\sum_{i=1}^{n}\frac{\pi(y_i|x_i)}{\pi_0(y_i|x_i)}$ with (x_i, y_i) draw i.i.d from $P(\mathcal{X})\times\pi_0(\mathcal{Y}|x)$. Also, it is easy to see $\mathbb{E}_{x,y\sim\pi_0(\cdot|x)}[\frac{\pi(y|x)}{\pi_0(y|x)}] = 1 - \mathcal{D}_{\mathcal{X}}(\pi|\pi_0)$. Let p_{min} denote the smallest propensity under supported action set with $p_{min} := \min_{x,y\in\mathcal{U}(x,\pi_0)^c}\pi_0(y|x) > 0$, then the random variable $\frac{\pi(y|x)}{\pi_0(y|x)}$ is strictly bounded between $[0, \frac{1}{p_{min}}]$. Applying Hoeffding's bound gives:

$$\begin{aligned}\mathbb{P}(\mathcal{D}_{\mathcal{X}}(\pi|\pi_0) < 1 - S_{\mathcal{D}}(\pi|\pi_0) - \epsilon) &= \mathbb{P}(S_{\mathcal{D}}(\pi|\pi_0) \\ &\quad - (1 - \mathcal{D}_{\mathcal{X}}(\pi|\pi_0)) < -\epsilon) \\ &\leq exp(-2n\epsilon^2 p_{min}^2)\end{aligned} \tag{23}$$

Since $S_{\mathcal{D}(\pi|\pi_0)} \leq 1 - \epsilon$ gives $1 - S_{\mathcal{D}}(\pi|\pi_0) - \epsilon \geq 0$, then we have

$$\mathbb{P}(\mathcal{D}_{\mathcal{X}}(\pi|\pi_0) < 0) \leq exp(-2n\epsilon^2 p_{min}^2) \tag{24}$$

Similar for the other direction, Hoeffding's bound gives:

$$\begin{aligned}\mathbb{P}(\mathcal{D}_{\mathcal{X}}(\pi|\pi_0) > 1 - S_{\mathcal{D}}(\pi|\pi_0) + \epsilon) &= \mathbb{P}(S_{\mathcal{D}}(\pi|\pi_0) - (1 - \mathcal{D}_{\mathcal{X}}(\pi|\pi_0)) \\ &\quad > \epsilon) \leq exp(-2n\epsilon^2 p_{min}^2)\end{aligned} \tag{25}$$

Since $S_{\mathcal{D}(\pi|\pi_0)} \geq 1 + \epsilon - \kappa$ gives $1 - S_{\mathcal{D}}(\pi|\pi_0) + \epsilon \leq \kappa$, then we have

$$\mathbb{P}(\mathcal{D}_{\mathcal{X}}(\pi|\pi_0) \geq \kappa) \leq exp(-2n\epsilon^2 p_{min}^2) \tag{26}$$

Combining the above, we have

$$\begin{aligned}\mathbb{P}(0 \leq \mathcal{D}_{\mathcal{X}}(\pi|\pi_0) \leq \kappa) &= 1 - \mathbb{P}(\mathcal{D}_{\mathcal{X}}(\pi|\pi_0) < 0) - \mathbb{P}(\mathcal{D}_{\mathcal{X}}(\pi|\pi_0) > \kappa) \\ &\geq 1 - 2exp(-2n\epsilon^2 p_{min}^2)\end{aligned} \tag{27}$$

□

B APPENDIX: EXPERIMENT DETAILS

Datasets and baseline. We follow a 75:10:15 train-validation-test split for credit card fraud detection dataset, while for CIFAR10 already coming with a train-test split, we keep 10% of the training set as validation set. Baseline estimators are IPS and DM, the hyperparameters (learning rate, L_2 regularization) are optimized for all the methods based on the validation set.

Bandit data generation. For CIFAR10, given supervised data in the format of $\{x_i, y_i^*\}_{i=1}^n$ where x_i denotes the 3072 features and y_i^* denotes the correct label of data (ranging from 0 to 9), under logging policy π_0, the logged bandit data is generated by drawing $y_i \sim \pi_0(\mathcal{Y}|x_i)$, then a deterministic reward is defined as $\mathbb{1}_{\{y_i=y_i^*\}}$. For the credit card fraud detection dataset, we aim to have continuous rewards rather than binary. To do so, we throw away the class label and only use the data features for each sample to generate bandit data. To be specific, for each sample with a 28-dimensional feature vector, we define the first 20 features as the contextual information, and use the remaining 8 features as the underlying true reward for 8 different actions (with normalization).

Logging policy. For CIFAR, we learn the softmax logging policy on 35K full-information data points as a multi-class classification problem with cross-entropy loss. Similar as the experiments on BanditNet [11], we adopt the conventional ResNet20 architecture but restrict training after a mere two epochs to derive a relative stochastic policy, since it will be easier to add temperature later to control its stochasticity and support deficiency. Similarly, for the credit card fraud detection dataset, the softmax logging policy is learned on 8K full-information data points by treating it as a multi-class classification problem using cross-entropy loss and the label being the action with the highest reward on this specific context. For CIFAR, the logging policy we trained has a 57.43% accuracy on the test-set; whereas for the credit card fraud detection dataset, the logging policy has an expected true reward of 0.71.

Reward estimator. For each experiment, we train a different regression function using the full bandit dataset. We use the same architecture as the one used for off-policy learning (ResNet20 for CIFAR10, two layer neural network for the Credit Card dataset) - where the final layer is the size of the actions, specifying the reward for each action given a particular context. The regression function is trained using the MSE objective.

Model selection details. We provide the model selection result for credit card in Table 2. In this dataset, both DM and MinSup achieve near-oracle performance under different levels of support deficiency. In Figure 6, we test how various estimators perform under model-misspecification on the CIFAR dataset. As the number of input feature decreases. the quality of DM diminishes, which affects its performance used in model selection. MinSup is pretty robust under model misspecification.

% Unsupp.	Oracle	DM	Cons. Extra.	MinSup
0	0.793	**0.793**	**0.793**	**0.793**
20	0.803	0.791	0.791	**0.803**
45	0.799	0.798	0.795	**0.799**
60	0.797	**0.797**	**0.797**	0.796
75	0.778	**0.778**	0.766	**0.778**
80	0.774	0.759	0.751	0.759

Table 2: Model selection results for Credit Card with varying levels of support deficiency in the logging policy.

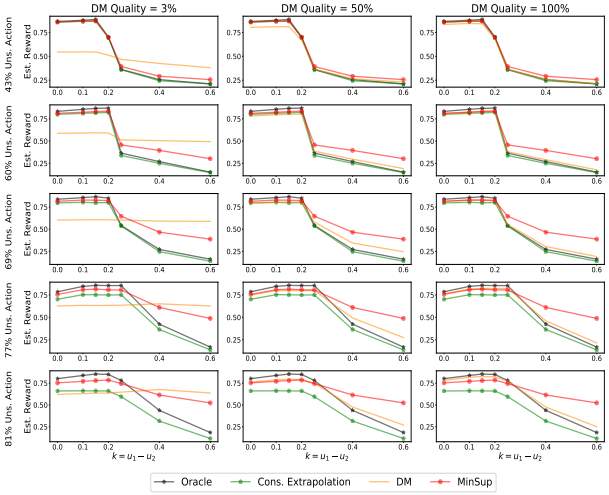

Figure 6: Model selection results for various support deficiencies and different levels of model misspecifications on CIFAR.