Machine Learning (2019) 108:945–970
https://doi.org/10.1007/s10994-018-5768-3

Constructing effective personalized policies using counterfactual inference from biased data sets with many features

Onur Atan[1] · William R. Zame[1,2] · Qiaojun Feng[3] · Mihaela van der Schaar[1,4]

Received: 15 December 2016 / Accepted: 3 October 2018 / Published online: 5 December 2018

Abstract
This paper proposes a novel approach for constructing effective personalized policies when the observed data lacks counter-factual information, is biased and possesses many features. The approach is applicable in a wide variety of settings from healthcare to advertising to education to finance. These settings have in common that the decision maker can observe, for each previous instance, an array of features of the instance, the action taken in that instance, and the reward realized—but not the rewards of actions that were not taken: the counterfactual information. Learning in such settings is made even more difficult because the observed data is typically biased by the existing policy (that generated the data) and because the array of features that might affect the reward in a particular instance—and hence should be taken into account in deciding on an action in each particular instance—is often vast. The approach presented here estimates propensity scores for the observed data, infers counterfactuals, identifies a (relatively small) number of features that are (most) relevant for each possible action and instance, and prescribes a policy to be followed. Comparison of the proposed algorithm against state-of-art algorithms on actual datasets demonstrates that the proposed algorithm achieves a significant improvement in performance.

Editor: Csaba Szepesvari.

✉ Onur Atan
oatan@ucla.edu

William R. Zame
zame@econ.ucla.edu

Qiaojun Feng
fqj13@mails.tsinghua.edu.cn

Mihaela van der Schaar
mihaela.vanderschaar@eng.ox.ac.uk

1 University of California, Los Angeles, Los Angeles, USA

2 Nuffield College, Oxford University, Oxford, UK

3 Tsinghua University, Beijing, China

4 Oxford-Man Institute, Oxford University, Oxford, UK

Keywords Inferring counterfactuals · Identifying relevant features · Constructing personalized policies

1 Introduction

The "best" treatment for the current patient must be learned from the treatment(s) of previous patients. However, no two patients are ever *exactly* alike, so the learning process must involve learning the ways in which the current patient *is* alike to previous patients—i.e., has the same or similar features—and which of those features are *relevant* to the treatment(s) under consideration. This already complicated learning process is further complicated because the history of previous patients records only outcomes actually experienced from treatments actually received—not the outcomes that would have been experienced from alternative treatments—the *counterfactuals*. And this learning process is complicated still further because the treatments received by previous patients were (typically) chosen according to some protocol that might or might not be known but was almost surely not random—so the observed data is *biased*.

The same complications arise in many other settings. Which mode of advertisement would be most effective for a given product? Which materials would best promote learning/performance for a given student? Which investment strategy would yield higher returns or lower risk in a particular macroeconomic environment? As in the medical setting, choosing the "best" policy in these settings (and in others too numerous to mention) requires learning which features of each context are relevant for the decision/action at hand and learning about the consequences of decisions/actions not taken in previous contexts—the *counterfactuals*; such learning is especially complicated because the observed data may be biased (because it was created by an existing—perhaps less effective—policy) and because each observed instance and action may be informed by a vast array of features. (Counterfactuals are seldom seen in observed data. One possible way to obtain counterfactual information would be to conduct controlled experiments—but in many contexts, experimentation will be impractical or even impossible. Absent controlled experiments, counterfactuals must be *inferred*.)

This paper proposes a novel approach to addressing such problems. We construct an algorithm that learns a nonlinear policy to recommend an action for each (new) instance. During the training phase, our algorithm learns the action-dependent relevant features and then uses a feedforward neural network to optimize a nonlinear stochastic policy the output of which is a probability distribution over the actions given the relevant features. When we apply the trained algorithm to a new instance, we choose the action which has the highest probability. In the settings mentioned above our algorithm constructs: (in the medical context) a personalized plan of patient treatment; (in the advertising context) a product-specific plan of advertisement; (in the educational context) a student-specific plan of instruction; (in the financial context) a situationally-specific investment strategy. We use actual data to demonstrate that our algorithm is significantly superior to existing state-of-the-art algorithms. We emphasize that our methods and the algorithms we develop are widely applicable to an enormous range of settings, from healthcare to advertisement to education to finance to recommender systems to smart cities. (See Athey and Imbens (2015), Hoiles and van der Schaar (2016) and Bottou et al. (2013), for just a few examples.)

As we have noted, our methods and algorithms apply in many settings, each of which comes with specific features, actions and rewards. In the medical context, typical features are items available in the electronic health record (laboratory tests, previous diagnoses, demographic

Table 1 Success rates of two treatments for kidney stones (Bottou et al. 2013)

	Overall	Small stones	Large stones
Open surgery	78%(273/350)	**93%(81/87)**	**73%(192/263)**
Percutaneous nephrolithotomy	**83%(289/350)**	87%(234/270)	69%(55/80)

Bold values indicate "better" performance

information, etc.), typical actions are choices of treatments (perhaps including no treatment at all), and typical rewards are recovery rates or 5-year survival rates. In the advertising context, typical features are the characteristics of a particular website and user, typical actions are the placements of an advertisement on a webpage, and typical rewards are click-rates. In the educational context, typical features are previous coursework and grades, typical actions are materials presented or subsequent courses taken, and typical rewards are final grades or graduation rates. In the financial context, typical features are aspects of the macroeconomic environment (interest rates, stock market information, etc.), typical actions are the timing of particular investment choices, and typical rewards are returns on investment.

For a simple but striking example from the medical context, consider the problem of choosing the best treatment for a patient with kidney stones. Such patients are usually classified by the size of the stones: small or large; the most common treatments are Open Surgery and Percutaneous Nephrolithotomy. Table 1 summarizes the results. Note that Open Surgery performs better than Percutaneous Nephrolithotomy for patients with small stones *and* for patients with large stones but Percutaneous Nephrolithotomy performs better overall.[1] Of course this would be impossible if the subpopulations that received the two treatments were identical—but they were not. And in fact we do not know the policy that created these subpopulations by assigning patients to treatments. We do know that patients are distinguished by a vast array of features in addition to the size of stones—age, gender, weight, kidney function tests, etc.—but we do not know which of these features is relevant. And of course we know the result of the treatment actually received by each patient—but we do not know what the result of the alternative treatment would have been (the counterfactual).

Three more points should be emphasized. Although Table 1 shows only two actions, in fact there are a number of other possible actions for kidney stones: they could be treated using any of a number of different medications, they could be treated by ultrasound, or they could not be treated at all. This is important for several reasons. The first is that a number of existing methods assume that there are only two actions (corresponding to treat or not-treat); but as this example illustrates, in many contexts (and in the medical context in particular), it is *typically* the case that there are *many* actions, not just two—and, as the papers themselves note, these methods simply do not work when there are more than two actions; see Johansson et al. (2016). The second is that the features that are relevant for predicting the success of a particular action typically depend on the action: different features will be found to be relevant for different actions. (The treatment of breast cancer, as discussed in Yoon et al. (2017), illustrates this point well. The issue is not simply whether or not to apply a regime of chemotherapy, but *which* regime of chemotherapy to apply. Indeed, there are at least six widely used regimes of chemotherapy to treat breast cancer, and the features that are relevant for predicting success of a given regime are different for different regimes.) The third is that we go much further than the existing literature by allowing for *nonlinear* policies. To do this, we use a feedforward neural network, rather than relying on familiar algorithms such

[1] This is a particular instance of Simpson's Paradox.

as POEM (Swaminathan and Joachims 2015a). To determine the best treatment, the bias in creating the populations, the features that are relevant *for each action* and the policy must all be *learned*. Our methods are adequate to this task.

The remainder of the paper is organized as follows. In Sect. 2, we describe some related work and highlight the differences with respect to our work. In Sect. 3, we describe the observational data on which our algorithm operates. In Sect. 4, we begin with an informal overview, then give the formal description of our algorithm (including substantial discussion). Section 5 gives the pseudo-code for the algorithm. Some extensions are discussed in Sect. 6. In Sect. 7, we demonstrate the performance of our algorithm on a variety of real datasets. Section 8 concludes. Proofs are in the Appendix.

2 Related work

From a conceptual point of view, the paper most closely related to ours—at least among recent papers—is perhaps Johansson et al. (2016) which treats a similar problem: learning relevance in an environment in which the counterfactuals are missing, data is biased and each instance may have many features. The approach taken there is somewhat different from ours in that, rather than identifying the relevant features, they transfer the features to a new representation space. [This process is referred as *domain adaptation* (Johansson et al. 2016).] A more important difference from our work is that it assumes that there are only two actions: treat and don't treat. As we have discussed in the Introduction, the assumption of two actions is unrealistic; in most situations there will be *many* (possible) actions. It states explicitly that the approach taken there does not work when there are more than two actions and offers the multi-action setting as an obvious but difficult challenge. One might think of our work as "solving" this challenge—but we stress that the "solution" is not at all a routine extension. Moreover, in addition to this obvious challenge, there is a more subtle—but equally difficult—challenge: when there are more than two actions, it will typically be the case that some features will be relevant for some actions and not for others, and—as discussed in the Introduction—it will be crucial to learn which features are relevant for which actions.

From a technical point of view, our work is perhaps most closely related to Swaminathan and Joachims (2015a) in that we use similar methods (IPS-estimates and empirical Bernstein inequalities) to learn counterfactuals. However, it does not treat observational data in which the bias is unknown and does not learn/identify relevant features. Another similar work on policy optimization from observational data is Strehl et al. (2010).

The work in Wager and Athey (2015) treats the related (but somewhat different) problem of estimating individual treatment effects. The approach there is through causal forests as developed by Athey and Imbens (2015), which are variations on the more familiar random forests. However, the emphasis in this work is on asymptotic estimates, and in the many situations for which the number of (possibly) relevant features is large the datasets will typically not be large enough that asymptotic estimates will be of more than limited interest. There are many other works focusing on estimating treatment effects; some include Tian et al. (2012), Alaa and van der Schaar (2017), Shalit et al. (2016).

More broadly, our work is related to methods for feature selection and counterfactual inference. The literature on feature selection can be roughly divided into categories according to the extent of supervision: supervised feature selection (Song et al. 2012; Weston et al. 2003), unsupervised feature selection (Dy and Brodley 2004; He et al. 2005) and semi-supervised feature selection (Xu et al. 2010). However, our work does not fall into any of

these categories; instead we need to select features that are informative in determining the rewards of each action. This problem was addressed in Tekin and van der Schaar (2014) but in an *online* Contextual Multi-Armed Bandit (CMAB) setting in which experimentation is used to learn relevant features. In the present paper, we treat the *logged* CMAB setting in which experimentation is impossible and relevant features must be learned from the existing logged data. As we have already noted, there are many circumstances in which experimentation is impossible. The difference between the settings is important—and the logged setting is much more difficult—because in the online setting it is typically possible to *observe* counterfactuals, while in the current logged setting it is typically *not* possible to observe counterfactuals, and because in the online setting the decision-maker controls the observations so whatever bias there is in the data is known.

With respect to learning, feature selection methods can be divided into three categories—filter models, wrapper models, and embedded models (Tang et al. 2014). Our method is most similar to filter techniques in which features are ranked according to a selected criterion such as a Fisher score (Duda et al. 2012), correlation based scores (Song et al. 2012), mutual information based scores (Koller and Sahami 1996; Yu and Liu 2003; Peng et al. 2005), Hilbert–Schmidt Independence Criterion (HSIC) (Song et al. 2012) and Relief and its variants (Kira and Rendell 1992; Robnik-Šikonja and Kononenko 2003) etc., and the features having the highest ranks are labeled as relevant. However, these existing methods are developed for classification problems and they cannot easily handle datasets in which the rewards of actions not taken are missing.

The literature on counterfactual inference can be categorized into three groups: direct, inverse propensity re-weighting and doubly robust methods. The direct methods compute counterfactuals by learning a function mapping from feature-action pair to rewards (Prentice 1976; Wager and Athey 2015). The inverse propensity re-weighting methods compute unbiased estimates by weighting the instances by their inverse propensity scores (Swaminathan and Joachims 2015a; Joachims and Swaminathan 2016). The doubly robust methods compute the counterfactuals by combining direct and inverse propensity score reweighing methods to compute more robust estimates (Dudík et al. 2011; Jiang and Li 2016). With respect to this categorization, our techniques might be view as falling into doubly robust methods.

Our work can be seen as building on and extending the work of Swaminathan and Joachims (2015a, b), which learn *linear* stochastic policies. We go much further by learning a *non-linear* stochastic policy. Our work can also be seen as an off-line variant of the on-line REINFORCE algorithm (Williams 1992).

We should also note two papers that were written *after* the current paper was originally submitted. The work of Joachims et al. (2018) extends the earlier work of Swaminathan and Joachims (2015a, b) to non-linear policies. Our own (preliminary) work (Atan et al. 2018) propose a different approach for learning a representation function and a policy. Unlike the present paper, our more recent work uses a loss function that embodies both a policy loss (similar to, but slightly different than, the policy loss used in the present paper) *and* a domain loss (which quantifies the divergence between the logging policy and the uniform policy under the representation function). The advantage of these changes is that they make it possible to learn the representation function and the policy in an end-to-end fashion.

3 Data

We consider logged contextual bandit data: that is, data for which we know the features of each instance, the action taken and the reward realized in that instance—but not the reward that would have been realized had a different action been taken. We assume that the data has been logged according to some policy *which we may not know, but which is not necessarily random* and so the data is *biased*. Each data point consists of a feature, an action and a reward. A *feature* is a vector $(x_1, \ldots, x_d)$ where each $x_i \in \mathcal{X}_i$ is a *feature type*. The space of all feature types is $\mathcal{F} = \{1, \ldots, d\}$, the space of all features is $\mathcal{X} = \Pi_{i=1}^{d} \mathcal{X}_i$ and the set of *actions* is $\mathcal{A}$. We assume that the sets of feature types and actions are finite; we write $b_i = |\mathcal{X}_i|$ for the cardinality of $\mathcal{X}_i$ and $\mathcal{A} = \{1, 2, \ldots, k\}$ for the set of actions. For $\boldsymbol{x} \in \mathcal{X}$ and $\mathcal{S} \subset \mathcal{F}$ we write $\boldsymbol{x}_{\mathcal{S}}$ for the restriction of $\boldsymbol{x}$ to $\mathcal{S}$; i.e. for the vector of feature types whose indices lie in $\mathcal{S}$. It will be convenient to abuse notation and view $\boldsymbol{x}_{\mathcal{S}}$ both as a vector of length $|\mathcal{S}|$ or as a vector of length $d = |\mathcal{F}|$ which is 0 for feature types not in $\mathcal{S}$. A *reward* is a real number; we normalize so that rewards lie in the interval [0, 1]. In some cases, the reward will be either 1 or 0 (success or failure; good or bad outcome); in other cases the reward may be interpreted as the probability of a success or failure (good or bad outcome).

We are given a data set

$$\mathcal{D}^n = \{(\boldsymbol{X}_1, A_1, R_1^{\text{obs}}), \ldots, (\boldsymbol{X}_n, A_n, R_n^{\text{obs}})\}$$

We assume that the jth instance/data point $(\boldsymbol{X}_j, A_j, R_j^{\text{obs}})$ is generated according to the following process:

1. The instance is described by a feature vector $\boldsymbol{X}_j$ that arrives according to the fixed but unknown distribution $\Pr(\mathcal{X})$; $\boldsymbol{X}_j \sim \Pr(\mathcal{X})$.
2. The action taken was determined by a policy that draws actions at random according to a (possibly unknown) probability distribution $p_0(\mathcal{A}|\boldsymbol{X}_j)$ on the action space $\mathcal{A}$. (Note that the distribution of actions taken depends on the vector of features).
3. Only the reward of the action actually performed is recorded into the dataset, i.e., $R_j^{\text{obs}} \equiv R_j(A_j)$.
4. For every action a, either taken or not taken, the reward $R_j(a) \sim \Phi_a(\cdot|\boldsymbol{X}_j)$ that would have been realized had a actually been taken is generated by a random draw from an unknown family $\{\Phi_a(\cdot|\boldsymbol{x})\}_{\boldsymbol{x}\in\mathcal{X}, a\in\mathcal{A}}$ of reward distributions with support [0, 1].

The logging policy corresponds to the choices made by the existing decision-making procedure and so will typically create a biased distribution on the space of feature-action pairs.

We make two natural assumptions about the rewards and the logging policy; taken together they enable us to generate unbiased estimates of the variables of the interest. The first assumption guarantees that there is enough information in the data-generating process so that counterfactual information can be inferred from what is actually observed.

Assumption 1 (Common support) $p_0(a|\boldsymbol{x}) > 0$ for all action-feature pairs $(a, \boldsymbol{x})$.

The second assumption is that the logging policy depends only on the observed features—and not on the observed rewards.

Assumption 2 (Unconfoundness) For each feature vector $\boldsymbol{X}$, the rewards of actions $\{R(a)\}_{a\in\mathcal{A}}$ are statistically independent of the action actually taken; $\{R(a)\} \perp\!\!\!\perp A \big| \boldsymbol{X}$.

These assumptions are universal in the counterfactual inference literature—see Johansson et al. (2016), Athey and Imbens (2015) for instance—although they can be criticized on the grounds that their validity cannot be determined on the basis of what is actually observed.

4 The algorithm

It seems useful to begin with a brief overview; more details and formalities follow below. Our algorithm consists of a training phase and an execution phase; the training phase consists of three steps.

A. In the first step of the training phase, the algorithm either inputs the true propensity scores (if they are known) or uses the logged data to estimate propensity scores (when the true propensity scores are not known); this (partly) corrects the bias in the logged data.
B. In the second step of the training phase, the algorithm uses the known or estimated propensity scores to compute, for each action and each feature, an estimate of relevance for that feature with respect to that action. The algorithm then retains the more relevant features—those for which the estimate is above a threshold—and discards the less relevant features—those for which the estimate is below the threshold. (For reasons that will be discussed below, the threshold used depends on both the action and the feature type.)
C. In the third step of the training phase, the algorithm uses the known or estimated propensity scores and the features identified as relevant, and trains a feedforward neural network model to learn a non-linear stochastic policy that minimizes the "corrected" cross entropy loss.

In the execution phase, the algorithm is presented with a new instance and uses the policy derived in the training phase to recommend an action for this new instance on the basis of the relevant features of that instance.

Not surprisingly, the setting in which the propensity scores are known is simpler than the setting in which the propensity scores must be estimated. In the latter case, in addition to the complication of the estimation itself, we shall need to be careful about estimated propensity scores that are "too small"—this will require a correction—and our error estimates will be less good. Because clarity of exposition seems more importance than compactness, we therefore present first the algorithm for the case in which true propensity scores are known and then circle back to present the necessary modifications for the case in which true propensity scores are not known but must be estimated.

4.1 True propensities

We begin with the setting in which propensities of the randomized algorithm are actually tracked and available in the dataset. This is often the case in the advertising context, for example. In this case, for each j, set $p_{0,j} = p_0(A_j|X_j)$, and write $\boldsymbol{P}_0 = [p_{0,j}]_{j=1}^n$; this is the vector of *true propensities*.

4.2 Relevance

It might seem natural to define the set $\mathcal{S}$ of feature types to be *irrelevant* (for a particular action) if the distribution of rewards (for that action) is independent of the features in $\mathcal{S}$, and to define the set $\mathcal{S}$ to be *relevant* otherwise. In theoretical terms, this definition has much to recommend it. In operational terms, however, this definition is not of much use. That is because finding irrelevant sets of feature types would require many observations (to determine the entire distribution of rewards) and intractable calculations (to examine all sets of feature types). Moreover, this notion of irrelevance will often be too strong because our interest will often be only in maximizing expected rewards (or more generally some statistical function

of rewards), as it would be in the medical context if the reward is five-year survival rate, or in the advertising or financial settings, if the reward is expected revenue or profit and the advertiser or firm is risk-neutral.

Given these objections, we take an alternative approach. We define a measure of how relevant a particular feature type is for the expected reward of a particular action, learn/estimate this measure from observed data, retain features for which this measure is above some endogenously derived threshold (the most relevant features) and discard other features (the least relevant features). Of course, this approach has drawbacks. Most obviously, it might happen that two feature types are individually not very relevant but are jointly quite relevant. (We leave this issue for future work.) However, as we show empirically, this approach has the virtue that it works: the algorithm we develop on the basis of this approach is demonstrably superior to existing algorithms.

4.2.1 True relevance

To begin formalizing our measure of relevance, fix an action a, a feature vector x and a feature type i. Define expected rewards and marginal expected rewards as follows:

$$\begin{aligned} \bar{r}(a, \boldsymbol{x}) &= \mathbb{E}\left[R(a) | \boldsymbol{X} = \boldsymbol{x}\right] \\ \bar{r}(a, \boldsymbol{x}_i) &= \mathbb{E}_{\boldsymbol{X}_{-i}}[\bar{r}(a, \boldsymbol{X}) \Big| \boldsymbol{X}_i = \boldsymbol{x}_i] \\ \bar{r}(a) &= \mathbb{E}_{\boldsymbol{X}}\left[\bar{r}(a, \boldsymbol{X})\right] \end{aligned} \tag{1}$$

We define the *true relevance of feature type i for action a* by

$$g(a, i) = \mathbb{E}\left[\ell\left(\bar{r}(a, X_i) - \bar{r}(a)\right)\right], \tag{2}$$

where the expectation is taken with respect to the arrival probability distribution of X_i and $\ell(\cdot)$ denotes the loss metric. (Keep in mind that the true arrival probability distribution of X_j is unknown and must be estimated from the data.) Our results hold for an arbitrary loss function, assuming only that it is strictly monotonic and Lipschitz; i.e. there is a constant B such that $\left|\ell(r) - \ell(r')\right| \leq B|r - r'|$. These conditions are satisfied by a large class of loss functions including l_1 and l_2 losses. The relevance measure g expresses the weighted difference between the expected reward *of a given action* conditioned on the feature type i and the unconditioned expected reward; $g(a, i) = 0$ exactly when feature type i does not affect the expected reward of action a.[2]

We refer to g as *true* relevance because it is computed using the *true* arrival distribution—but the true arrival distribution is unknown. Hence, even when the true propensities are known, relevance must be *estimated* from observed data. This is the next task.

4.2.2 Estimated relevance

We now derive *estimates* of relevance based on observed data (continuing to assume that true propensities are known). To do so, we first need to estimate $\bar{r}(a)$ and $\bar{r}(a, x_i)$ for $x_i \in \mathcal{X}_i$, $i \in \mathcal{F}$ and $a \in \mathcal{A}$ from available observational data. An obvious way to do this is through classical supervised learning based estimators; most obviously, the sample mean estimators for $\bar{r}(a)$ and $\bar{r}(a, x_i)$. However using straightforward sample mean estimation would be

[2] Other measures of relevance have been used in the feature selection literature [e.g., especially Pearson correlation (Hall 1999) and mutual information (Yu and Liu 2003)]—but not for relevance of actions.

wrong because the logging policy introduces a bias into observations. Following the idea of Inverse Propensity Scores (Rosenbaum and Rubin 1983), we correct this bias by using Importance Sampling.

Write $N(a)$, $N(x_i)$, $N(a, x_i)$ for the number of observations (in the given data set) with action a, with feature x_i, and with the pair consisting of action a and feature x_i, respectively. We can rewrite our previous definitions as:

$$\begin{aligned}\bar{r}(a, x_i) &= \mathbb{E}_{(\boldsymbol{X}, A, R^{\text{obs}})\sim p_0}\left[\frac{\mathbb{I}(A = a)R^{\text{obs}}}{p_0(A|\boldsymbol{X})}\middle| X_i = x_i\right] \\ \bar{r}(a) &= \mathbb{E}_{(\boldsymbol{X}, A, R^{\text{obs}})\sim p_0}\left[\frac{\mathbb{I}(A = a)R^{\text{obs}}}{p_0(A|\boldsymbol{X})}\right]\end{aligned} \tag{3}$$

where $\mathbb{I}(\cdot)$ is the indicator function. (Note that we are taking expectations with respect to the true propensities.)

Let $\mathcal{J}(x_i)$ denote the time indices in which feature type-i is x_i, i.e., $\mathcal{J}(x_i) = \{j \subseteq \{1, 2, \ldots, n\} : X_{i,j} = x_i\}$. The Importance Sampling approach provides unbiased estimates of $\bar{r}(a)$ and $\bar{r}(a, x_i)$ as

$$\begin{aligned}\widehat{R}(a, x_i; \boldsymbol{P}_0) &= \frac{1}{N(x_i)} \sum_{j\in\mathcal{J}(x_i)} \frac{\mathbb{I}(A_j = a)R_j^{\text{obs}}}{p_{0,j}}, \\ \widehat{R}(a; \boldsymbol{P}_0) &= \frac{1}{n} \sum_{j=1}^{n} \frac{\mathbb{I}(A_j = a)R_j^{\text{obs}}}{p_{0,j}},\end{aligned} \tag{4}$$

(We include the propensities $\boldsymbol{P}_0$ in the notation as a reminder that these estimators are using the *true* propensity scores.)

We now define the *estimated relevance of feature type i for action a* as

$$\widehat{G}(a, i; \boldsymbol{P}_0) = \frac{1}{n} \sum_{x_i\in\mathcal{X}_i} N(x_i)\ell\left(\widehat{R}(a, x_i; \boldsymbol{P}_0) - \widehat{R}(a; \boldsymbol{P}_0)\right). \tag{5}$$

(Note that we have abused terminology/notation by suppressing reference to the particular sample that was observed.)

4.2.3 Thresholds

By definition, $\widehat{G}$ is an estimate of relevance so the obvious way to select relevant features is to set a threshold τ, identify a feature i as relevant for action a exactly when $\widehat{G}(a, i; \boldsymbol{P}_0) > \tau$, retain the features that are relevant according to this criterion and discard other features.

However, this approach is a bit too naive for (at least) two reasons. The first is that our empirical estimate of relevance $\widehat{G}$ may in fact be far from the true relevance g. The second is that some features may be highly (positively or negatively) correlated with the remaining features, and hence convey less information. To deal with these objections, we construct thresholds $\tau(a, i)$ as a weighted sum of an empirical estimate of the error in using $\widehat{G}$ instead of g and the (average absolute) correlation of feature type i with other feature types.

To define the first term we need an empirical (data-dependent bound) on $|\widehat{G} - g|$. To derive such a bound we use the empirical Bernstein inequality (Maurer and Pontil 2009; Audibert et al. 2009). (We emphasize that our bound depends on the *empirical variance* of the estimates.) To simplify notation, define random variables $U(a; \boldsymbol{P}_0) \equiv \frac{\mathbb{I}(A=a)R^{\text{obs}}}{p_0(A|\boldsymbol{X})}$ and

$U_j(a;\boldsymbol{P}_0) \equiv \frac{\mathbb{I}(A_j=a)R_j}{p_{0,j}}$. The sample means and variances are:

$$\begin{aligned}
&\mathbb{E}_{(\boldsymbol{X},A,R^{\text{obs}})\sim p_0}[U(a;\boldsymbol{P}_0)] = \bar{r}(a),\\
&\mathbb{E}_{(\boldsymbol{X},A,R^{\text{obs}})\sim p_0}[U(a;\boldsymbol{P}_0)\big|X_i = x_i] = \bar{r}(a,x_i)\\
&\widehat{U}(a;\boldsymbol{P}_0) = \widehat{R}(a;\boldsymbol{P}_0)\\
&= \frac{1}{n}\sum_{j=1}^{n} U_j(a;\boldsymbol{P}_0),\\
&\widehat{U}(a,x_i;\boldsymbol{P}_0) = \widehat{R}(a,x_i;\boldsymbol{P}_0)\\
&= \frac{1}{N(x_i)}\sum_{j\in\mathcal{J}(x_i)} U_j(a;\boldsymbol{P}_0),\\
&V_n(a;\boldsymbol{P}_0) = \frac{1}{n-1}\sum_{j=1}^{n}\left(U_j(a;\boldsymbol{P}_0) - \widehat{U}(a;\boldsymbol{P}_0)\right)^2,\\
&V_n(a,x_i;\boldsymbol{P}_0) = \frac{1}{N(x_i)-1}\sum_{j\in\mathcal{J}(x_i)}\left(U_j(a;\boldsymbol{P}_0) - \widehat{U}(a,x_i;\boldsymbol{P}_0)\right)^2.
\end{aligned}$$

The weighted average sample variance is:

$$\bar{V}_n(a,i;\boldsymbol{P}_0) = \sum_{x_i\in\mathcal{X}_i}\frac{N(x_i)V_n(a,x_i;\boldsymbol{P}_0)}{n} \tag{6}$$

Our empirical (data-dependent) bound is given in Theorem 1.

Theorem 1 *For every $n > 0$, every $\delta \in \left[0, \frac{1}{3}\right]$, and every pair, $(a,i) \in (\mathcal{A},\mathcal{D})$, with probability at least $1-3\delta$ we have:*

$$\begin{aligned}
|\widehat{G}(a,i;\boldsymbol{P}_0) - g(a,i)| \le B\Bigg(&\sqrt{\frac{2b_i\ln(3/\delta)\bar{V}_n(a,i;\boldsymbol{P}_0)}{n}}\\
&+\sqrt{\frac{2\ln(3/\delta)V_n(a;\boldsymbol{P}_0)}{n}}\\
&+\frac{M\,(b_i+1)\ln 3/\delta}{n}\Bigg)\\
&+\sqrt{\frac{2\,(\ln 1/\delta + b_i\ln 2)}{n}},
\end{aligned}$$

where $M = \max_{a\in\mathcal{A}}\max_{\boldsymbol{x}\in\mathcal{X}} 1/p_0(a|\boldsymbol{x})$.

The error bound given by Theorem 1 consists of four terms: The first term arises from estimation error of $\widehat{R}(a,x_i)$. The second term arises from estimation error of $\widehat{R}(a)$. The third term arises from estimation error of feature arrival probabilities. The fourth term arises from randomness of the logging policy.

Now write $\rho_{i,j}$ for the Pearson correlation coefficient between two feature types i and j. (Recall that $\rho_{i,j} = +1$ if i,j are perfectly positively correlated, $\rho_{i,j} = -1$ if i,j are perfectly negatively correlated, and $\rho_{i,j} = 0$ if i,j are uncorrelated.) Then the average absolute correlation of feature type i with other features is

$$\left(\frac{1}{d-1}\right)\left(\sum_{j\in\mathcal{F}\setminus\{i\}}|\rho_{i,j}|\right)$$

We now define the thresholds as

$$\tau(a,i) = \lambda_1 \sqrt{\frac{b_i \bar{V}_n(a,i;\boldsymbol{P}_0)}{n}} + \lambda_2 \left(\frac{1}{d-1}\right)\left(\sum_{j \in \mathcal{F}\setminus\{i\}} |\rho_{i,j}|\right)$$

where λ_1, λ_2 are weights (hyper-parameters) to be chosen. Notice that the first term is the dominant term in the error bound given in Theorem 1, and is used to set a higher bar for the feature types that are creating the logging policy bias. The statistical distributions of those features within the the action population and the whole population will be different. By setting the threshold as above, we trade-off between three objective: (1) selecting the features that are relevant for the rewards of the actions, (2) eliminating the features which create the logging policy bias, (3) minimizing the redundancy in the feature space.

4.2.4 Relevant feature types

Finally, we identify the set of feature types that are relevant for an action a as

$$\widehat{\mathcal{R}}(a) = \left\{ i \ : \ \widehat{G}(a,i;\mathbf{P}_0) > \tau(a,i) \right\} \tag{7}$$

Set $\widehat{\boldsymbol{\mathcal{R}}} = \left[\widehat{\mathcal{R}}(a)\right]_{a\in\mathcal{A}}$. Let $\boldsymbol{f}_a$ denote a d dimensional vector whose jth element is 1 if j is contained in the set $\mathcal{R}(a)$ and 0 otherwise.

4.3 Policy optimization

We now build on the identified family of relevant features to construct a policy. By definition, a (stochastic) policy is a map $h : \mathcal{X} \to \triangle(A)$ which assigns to each vector of features a probability distribution $h(\cdot|\boldsymbol{x})$ over actions.

A familiar approach to the construction of stochastic policies is to use the POEM algorithm (Swaminathan and Joachims 2015a). POEM considers only linear stochastic policies; among these, POEM learns one that minimizes risk, adjusted by a variance term. Our approach is substantially more general because we consider arbitrary non-linear stochastic policies. We use a novel approach that uses a feedforward neural network to find a non-linear policy that minimizes the loss, adjusted by a regularization term. Note that we allow for very general loss and regularization terms so that our approach includes many policy optimizers. If we restricted to a neural network with no hidden layers and a specific regularization term, we would recover POEM.

We propose a feedforward neural network for learning a policy $h^*(\cdot|\boldsymbol{x})$; the architecture of our neural network is depicted in Fig. 1. Our feedforward neural network consists of policy layers (L_p hidden layers with $h_p^{(l)}$ units in the lth layer) that use the output of the concatenation layer to generate a policy vector $\Phi(\boldsymbol{x},a)$, and a softmax layer that turns the policy vector into a stochastic policy.

For each action a, the concatenation layer takes the feature vector $\boldsymbol{x}$ as an input and generates a action-specific representations $\phi(\boldsymbol{x},a)$ according to:

$$\begin{aligned} \boldsymbol{x}_{\widehat{\mathcal{R}}(a)} &= \boldsymbol{x} \odot \boldsymbol{f}_a \\ \phi(\boldsymbol{x},a) &= [\boldsymbol{x}_{\widehat{\mathcal{R}}(\tilde{a})} \mathbb{I}(\tilde{a} = a)]_{\tilde{a}\in\mathcal{A}} \end{aligned}$$

Note that our action-specific representation $\phi(\boldsymbol{x},a)$ is a $d \times k$ dimensional vector where only the parts corresponding to action a is non-zero and equals to $\boldsymbol{x}_{\widehat{\mathcal{R}}(\tilde{a})}$. For each action a, the

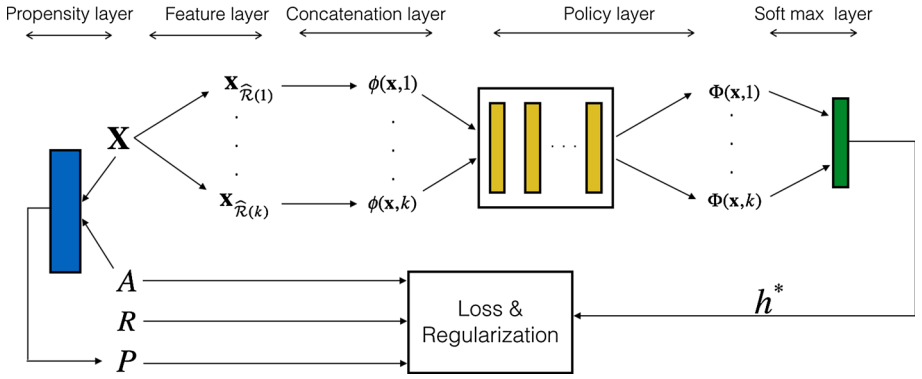

Fig. 1 Neural network architecture

policy layers uses the action-specific representation $\phi(\boldsymbol{x}, a)$ generated by the concenation layers and generates the output vector $\Phi(\boldsymbol{x}, a)$ according to:

$$\Phi(\boldsymbol{x}, a) = \rho\left(\ldots \rho\left(\boldsymbol{W}_1^{(p)}\phi(\boldsymbol{x}, a) + \boldsymbol{b}_1^{(p)}\right)\ldots + \boldsymbol{b}_{L_p}^{(p)}\right)$$

where $\boldsymbol{W}_l^{(p)}$ and $\boldsymbol{b}_l^{(p)}$ are the weights and bias vectors of the lth layer accordingly. The outputs of the policy layers are used to generate a policy by a softmax layer:

$$h(a|\boldsymbol{x}) = \frac{\exp(\boldsymbol{w}^T\Phi(\boldsymbol{x}, a))}{\sum_{a'\in\mathcal{A}}\exp(\boldsymbol{w}^T\Phi(\boldsymbol{x}, a'))}.$$

Then, we choose the parameters of the policy to minimize an objective of the following form: $\text{Loss}(h^*; \mathcal{D}) + \lambda_3\mathcal{R}(h^*; \mathcal{D})$; where $\text{Loss}(h^*; \mathcal{D})$ is the loss term, $\mathcal{R}(h^*; \mathcal{D})$ is a regularization term and $\lambda_3 > 0$ represents the trade-off between loss and regularization. The loss function can be either the negative IPS estimate or the corrected cross entropy loss introduced in the next section. Depending on the choice of the loss function and regularizer, our policy optimizer can include a wide-range of objectives including the POEM objective (Swaminathan and Joachims 2015a).

In the next subsection, we propose a new objective, which we refer to as the Policy Neural Network (PONN) objective.

4.4 Policy neural network (PONN) objective

Our PONN objective is motivated by the cross-entropy loss used in the standard multi-class classification setting. In the usual classification setting, usual loss function used to train the neural network is the standard cross entropy:

$$\widehat{\text{Loss}}_c(h) = \frac{1}{n}\sum_{j=1}^{n}\sum_{a\in\mathcal{A}} -R_j(a)\log h(a|\boldsymbol{X}_j).$$

However, this loss function is not applicable in our setting, for two reasons. The first is that only the rewards of the action taken by the logging policy are recorded in the dataset, not the counterfactuals. The second is that we need to correct the bias in the dataset by weighting the instances by their inverse propensities. Hence, we use the following modified cross entropy loss function:

$$\widehat{\text{Loss}}_b(h;\boldsymbol{P}_0) = \frac{1}{n}\sum_{j=1}^{n}\sum_{a\in\mathcal{A}}\frac{-R_j(a)\log h(a|\boldsymbol{X}_j)\mathbb{I}(A_j=a)}{p_{0,j}}$$
$$= \frac{1}{n}\sum_{j=1}^{n}\frac{-R_j^{\text{obs}}\log h(A_j|\boldsymbol{X}_j)}{p_{0,j}}. \tag{8}$$

Note that this loss function is an unbiased estimate of the expected cross entropy loss, that is $\mathbb{E}_{(\boldsymbol{X},A,R)\sim p_0}\left[\widehat{\text{Loss}}_b(h^*;\boldsymbol{P}_0)\right] = \mathbb{E}\left[\widehat{\text{Loss}}_c(h^*)\right]$. We train our neural network to minimize the regularized loss by Adam optimizer:

$$h^* = \arg\min_{h\in\mathcal{H}} \widehat{\text{Loss}}_b(h;\widehat{\boldsymbol{P}}_0) + \lambda_3\mathcal{R}(h),$$

where $\mathcal{R}(h)$ is the regularization term to avoid overfitting and λ_3 is the hyperparameter to trade-off between the loss and regularization.

4.5 Unknown propensities

As we have noted, in most settings the logging policy is unknown and hence the actual propensities are also unknown so we must *estimate* propensities from the dataset and use the *estimated* propensities to correct the bias. In general, this can be accomplished by any supervised learning technique.

For our purposes we estimate propensities by fitting the multinomial logistic regression model:

$$\ln(\Pr(A=a)) = \boldsymbol{\beta}_{0,a}^T\boldsymbol{X} - \ln Z \tag{9}$$

where $Z = \sum_{a\in\mathcal{A}}\exp\left(\boldsymbol{\beta}_{0,a}^T\boldsymbol{X}\right)$. The estimated propensities are

$$\widehat{p}_{0,j} \equiv \frac{\exp(\boldsymbol{\beta}_{0,A_j}^T\boldsymbol{X}_j)}{Z_j}$$

where we have written $Z_j = \sum_{a\in\mathcal{A}}\exp(\boldsymbol{\beta}_{0,a}^T\boldsymbol{X}_j)$. Write $\widehat{\boldsymbol{P}}_0 = [\widehat{p}_{0,j}]_{j=1}^n$ for the vector of estimated propensities.

In principle, we could use these estimated propensities in place of known propensities and proceed exactly as we have done above. However, there are two problems with doing this. The first is that if the estimated propensities are very small (which might happen because the data was not completely representative of the true propensities), the variance of the estimate $\widehat{G}$ will be too large. The second is that the thresholds we have constructed when propensities are known may no longer be appropriate when propensities must be estimated.

To avoid the first problem, we follow Ionides (2008) and modify the estimated rewards by truncating the importance sampling weights. This leads to "truncated" estimated rewards as follows:

$$\widehat{R}_m(a,x_i;\widehat{\boldsymbol{P}}_0) = \frac{1}{N(x_i)}\sum_{j\in\mathcal{J}(x_i)}\min\left(\frac{\mathbb{I}(A_j=a)}{\widehat{p}_{0,j}},m\right)R_j^{\text{obs}},$$
$$\widehat{R}_m(a;\widehat{\boldsymbol{P}}_0) = \frac{1}{n}\sum_{j=1}^{n}\min\left(\frac{\mathbb{I}(A_j=a)}{\widehat{p}_{0,j}},m\right)R_j^{\text{obs}}.$$

Algorithm Training Phase of the Algorithm PONN-B

Input: $\lambda_1, \lambda_2, \lambda_3, L_r, L_p, h_i^r, h_j^a$
Step A: Estimate propensities using a logistic regression
Compute $\boldsymbol{\beta}_{0,a}$ for each a by training Logistic regression model from (9).
Set $\widehat{p}_{0,j} = \exp(\boldsymbol{\beta}_{0,A_j}^T \boldsymbol{X}_j)/Z_j$ with $Z_j = \sum_{a\in\mathcal{A}} \exp(\boldsymbol{\beta}_{0,a}^T \boldsymbol{X}_j)$.
Step B: Identify the relevant features
Compute $\widehat{R}(a, x_i; \widehat{\boldsymbol{P}}_0)$, $\widehat{R}(a; \widehat{\boldsymbol{P}}_0)$, $\bar{V}_n(a, i; \widehat{\boldsymbol{P}}_0)$, $\rho_{i,l}$ for each a, x_i, i, l.
Compute $\widehat{G}(a, i; \widehat{\boldsymbol{P}}_0)$ for each action-feature type pair.
Solve $\widehat{\mathcal{R}}(a)$ from (7).
Step C: Policy Optimization
while until convergence **do**
  $\left(\boldsymbol{w}, \boldsymbol{W}_p^{(l)}\right) \leftarrow \text{Adam}\left(\mathcal{D}^{(n)}, \boldsymbol{w}, \boldsymbol{W}_p^{(l)}\right)$
end while
Output of Training Phase: Policy h^*, Features $\widehat{\boldsymbol{\mathcal{R}}}$

Algorithm Execution Phase of the Algorithm PONN-B

Input: Instance with feature $\boldsymbol{X}$
Set $\hat{a}(\boldsymbol{X}) = \arg\max_{a\in\mathcal{A}} h^*(a|\boldsymbol{X})$
Output of Execution phase: Recommended action $\hat{a}(\boldsymbol{X})$

Given these "truncated" estimated rewards, we define a "truncated" estimator of relevance by

$$\widehat{G}_m(a, i; \widehat{\boldsymbol{P}}_0) = \sum_{x_i\in\mathcal{X}_i} \frac{N(x_i)}{n} l\left(\widehat{R}_m(a, x_i; \widehat{\boldsymbol{P}}_0) - \widehat{R}_m(a; \widehat{\boldsymbol{P}}_0)\right)$$

From this point on, we proceed exactly as before, using the "truncated" estimator $\widehat{G}_m$ instead of $\widehat{G}$.

Note that $\widehat{R}_m(a, x_i; \widehat{\boldsymbol{P}}_0)$ and $\widehat{R}_m(a; \widehat{\boldsymbol{P}}_0)$ are not unbiased estimators of $\bar{r}(a, x_i)$ and $\bar{r}(a)$. The bias is due to using estimated truncated propensity scores which may deviate from true propensities. Let $\text{bias}(\widehat{R}_m(a; \widehat{\boldsymbol{P}}_0))$ denote the bias of $\widehat{R}_m(a; \widehat{\boldsymbol{P}}_0)$, which is given by

$$\text{bias}(\widehat{R}_m(a; \widehat{\boldsymbol{P}}_0)) = \bar{r}(a) - \mathbb{E}\left[\widehat{R}_m(a; \widehat{\boldsymbol{P}}_0)\right].$$

In the Appendices, we show the effect of this bias on the learning process.

5 Pseudo-code for the algorithm PONN-B

Below, we provide the pseudo-code for our algorithm which we call PONN-B (because it uses the PONN objective and Step B) exactly as discussed above. The first three steps constitute the offline training phase; the fourth step is the online execution phase. Within the training phase the steps are: Step A: Input propensities (if they are known) or estimate them using a logistic regression (if they are not known). Step B: Construct estimates of relevance (truncated if propensities are estimated), construct thresholds (using given hyper-parameters) and identify the relevant features as those for which the estimated relevance is above the constructed thresholds. Step C: Use the Adam optimizer to train neural network

parameters. In the execution phase: Input the features of the new instance, apply the optimal policy to find a probability distribution over actions, and draw a random sample action from this distribution.

6 Extension: relevant feature selection with fine gradations

Our algorithm might be inefficient when there are many features of a particular type—in particular, if one or more feature types are continuous. In that setting, we can modify our algorithm to create bins that consist of *similar* feature values and treat all the values in a single bin identically. In order to conveniently formalize this problem, we assume that the feature space is actually continuous; for simplicity we assume each feature type is $\mathcal{X}_i = [0, 1]$ (or a bounded subset). In this case, we can partition the feature space into subintervals (bins), view features in each bin as identical, and apply our algorithm to the finite set of bins.[3] To offer a theoretical justification for this procedure, we assume that similar features yield similar expected rewards. We formalize this as a Lipschitz condition.

Assumption 3 There exists $L > 0$ such that for all $a \in \mathcal{A}$, all $i \in \mathcal{F}$ and all $x_i \in \mathcal{X}_i$, we have $|\bar{r}(a, x_i) - \bar{r}(a, \tilde{x}_i)| \leq L|x_i - \tilde{x}_i|$.

(In the Multi-Armed Bandit literature (Slivkins 2014; Tekin and van der Schaar 2014) this assumption is commonly made and sometimes referred to as *similarity*.)

For convenience, we partition each feature type X_i into s equal subintervals (bins) of length $1/s$. If s is small, the number of bins is small so, given a finite data set, the number of instances that lie in each bin is relatively large; this is useful for estimation. However, when s is small the size $1/s$ of each bin is relatively large so the (true) variation of expected rewards in each bin is relatively large. Because we are free to choose the parameter s, we can balance the trade-off implicit between choosing few large bins or choosing many small bins; a useful trade-off is achieved by taking $s = \lceil n^{1/3} \rceil$.

So begin by fixing $s = \lceil n^{1/3} \rceil$ and partition each $\mathcal{X}_i = [0, 1]$ into s intervals of length $1/s$. Write $\mathcal{C}_i$ for the sets in the partition of X_i and write c_i for a typical element of $\mathcal{C}_i$. For each sample j, let $c_{i,j}$ denote the set in which the feature $x_{i,j}$ belongs. Let $\mathcal{J}(c_i)$ be the set of indices for which $x_{i,j} \in c_i$; $\mathcal{J}(c_i) = \{j \in \{1, 2, \ldots, n\} : X_{i,j} \in c_i\}$. We define truncated IPS estimate as

$$\begin{aligned}
\bar{r}_m(a, c_i; \widehat{\boldsymbol{P}}_0) &= \mathbb{E}\left[U(a; \widehat{\boldsymbol{P}}_0) | X_i \in c_i\right] \\
&= \mathbb{E}\left[\min\left(\frac{\mathbb{I}(A = a)}{\widehat{p}_0(A|\boldsymbol{X})}, m\right) R^{\text{obs}} \middle| X_i \in c_i\right], \\
\widehat{R}_m(a, c_i; \widehat{\boldsymbol{P}}_0) &= \frac{1}{N(c_i)} \sum_{j \in \mathcal{J}(c_i)} \min\left(\frac{\mathbb{I}(A_j = a)}{\widehat{p}_{0,j}}, m\right) R_j^{\text{obs}},
\end{aligned}$$

where $N(c_i) = |\mathcal{J}(c_i)|$. In this case, we define estimated information gain as

$$\widehat{G}_m(a, i) = \sum_{c_i \in \mathcal{C}_i} \frac{N(c_i)}{n} l\left(\widehat{R}_m(a, c_i; \widehat{\boldsymbol{P}}_0) - \widehat{R}_m(a; \widehat{\boldsymbol{P}}_0)\right).$$

[3] The binning procedure loses the ordering in the interval [0, 1]. If this ordering is in fact relevant to the feature, then the binning procedure loses some information that a different procedure might preserve. We leave this for future work.

We define the following sample mean and variance :

$$\widehat{U}(a, c_i; \widehat{\boldsymbol{P}}_0) = \widehat{R}_m(a, c_i; \widehat{\boldsymbol{P}}_0) = \frac{1}{N(x_i)} \sum_{j \in \mathcal{J}(c_i)} U_j(a; \widehat{\boldsymbol{P}}_0),$$

$$V_n(a, c_i; \widehat{\boldsymbol{P}}_0) = \frac{1}{n-1} \sum_{j \in \mathcal{J}(c_i)} (U_j(a, c_i; \widehat{\boldsymbol{P}}_0) - \widehat{U}(a, c_i; \widehat{\boldsymbol{P}}_0))^2.$$

Let $\bar{V}_n(a, i; \widehat{\boldsymbol{P}}_0) = \sum_{c_i \in \mathcal{C}_i} \frac{N(c_i) V_n(a, c_i; \widehat{\boldsymbol{P}}_0)}{n}$ denote the weighted average sample variance.

Given these definitions, we establish a data-dependent bound analogous to Theorem 1.

Theorem 2 *For every $n \geq 1$ and $\delta \in \left[0, \frac{1}{3}\right]$, if $s = \lceil n^{1/3} \rceil$, then with probability at least $1 - 3\delta$ we have, for all pairs $(a, i) \in (\mathcal{A}, \mathcal{D})$,*

$$\begin{aligned} |\widehat{G}_m(a, i; \widehat{\boldsymbol{P}}_0) - g(a, i)| \leq B \Bigg(& \frac{\sqrt{4 \ln 3/\delta}}{n^{1/3}} \left(\sqrt{\bar{V}_n(a, i; \widehat{\boldsymbol{P}}_0)} + \sqrt{V_n(a; \widehat{\boldsymbol{P}}_0)} \right) + \frac{L}{n^{1/3}} \\ & + \left| \text{bias}(\widehat{R}_m(a; \widehat{\boldsymbol{P}}_0)) \right| + \mathbb{E} \left| \text{bias}(\widehat{R}_m(a, X_i; \widehat{\boldsymbol{P}}_0)) \right| \Bigg) \\ & + \frac{4mB \ln 3/\delta + \sqrt{2 \ln 1/\delta + \ln 2}}{n^{2/3}}. \end{aligned}$$

There are two main differences between Theorems 1 and 2. The first is that the estimation error is decreasing as $n^{1/3}$ (Theorem 2) rather than as $n^{1/2}$ (Theorem 1). The second is that there is an additional error in Theorem 2 arising from the Lipschitz bound.

Theorem 2 suggests a different choice of thresholds, namely:

$$\tau(a, i) = \lambda_1 n^{-1/3} \sqrt{V_n(a, i; \widehat{\boldsymbol{P}}_0)} + \lambda_2 \left(\frac{1}{d-1} \right) \left(\sum_{l \in \mathcal{F} \setminus \{i\}} |\rho_{i,l}| \right).$$

With this change we proceed exactly as before.

7 Numerical results

Here we describe the performance of our algorithm on some real datasets. Note that it is difficult (perhaps impossible) to validate and test the algorithm on the basis of actual logged CMAB data unless the counterfactual action rewards for each instance are available—which would (almost) never be the case. One way to validate and test our algorithm is to use a multi-class classification dataset, generate a biased CMAB dataset for training by "forgetting" (stripping out) the counterfactual information, apply the algorithm, and then test the predictions of the algorithm against the actual data (Beygelzimer et al. 2009). This is the route we follow in the first experiment below. Another way to validate and test our algorithm is to use an alternative accepted procedure to infer counterfactuals and to test the prediction of our algorithm against this alternative accepted procedure. This is the route we follow in the second experiment below.

Table 2 Data summary

Dataset	# of Feature types (d)	# of Labels (k)	# of Instances (n)
Pendigits	16	10	7494
Satimage	36	6	4435
Optdigits	64	10	3893

7.1 Multi-class classification

For this experiment we use existing multi-class classification datasets from the well-known UCI Machine Learning Repository.

- In the **Pendigits** and **Optdigits** datasets, each instance is described by a collection of pixels extracted from the image of a handwritten digit 0-9; the objective is to identify the digit from the features.
- In the **Satimage** dataset, each instance is described by an array of features extracted from a satellite image of a plot of ground; the objective is to identify the true description of the plot (barren soil, grass, cotton crop, etc.) from the features.

These datasets have in common that that they have many instances, many feature types and many labels, so they are extremely useful for training and testing.

In supervised learning systems, we assume that features and labels are generated by an i.i.d. process, i.e., $(\boldsymbol{X}, Y) \sim Z$ where $\boldsymbol{X} \in \mathcal{X}$ is the feature space and $Y \in \{1, 2, \ldots, k\}$ is the label space. The supervised learning data with n-samples is denoted as $\mathcal{D}^n = \left(\boldsymbol{X}_j, Y_j\right)_{j=1}^n$. In our simulation setup, we treat each class as an action. We also included 16 irrelevant features in addition to actual features in the dataset, drawn randomly from normal distribution. The reward of an action is given by $R_j(a) = \mathbb{I}(Y_j = a)$. A complete dataset then is $\mathcal{D}^n_{\text{com}} = \left(\boldsymbol{X}_j, R_j(1), \ldots, R_j(k)\right)$. A summary of the data is given in Table 2.

7.2 Comparisons

We compare the performance of our algorithm (PONN-B) with

- **PONN** is PONN-B but *without* Step B (feature selection).
- **POEM** is the standard POEM algorithm (Swaminathan and Joachims 2015a).
- **POEM-B** applies Step B of our algorithm, followed by the POEM algorithm.
- **POEM-L1** is the POEM algorithm with the addition of L_1 regularization.
- **Multilayer Perceptron with L_1 regularization (MLP-L1)** is the MLP algorithm on concatenated input $(\boldsymbol{X}, A)$ with L_1 regularization.
- **Logistic Regression with L_1 regularization (LR-L1)** is the separate LR algorithm on input $\boldsymbol{X}$ on each action a with L_1 regularization.
- **Logging** is the logging policy performance.

(In all cases, the objective is optimized with the Adam Optimizer.)

7.2.1 Simulation setup

We generate artificially biased dataset by the following logistic model. We first draw weights for each label from an multivariate Gaussian distribution, that is $\theta_{0,y} \sim \mathcal{N}(0, \kappa I)$. We then

use the logistic model to generate an artificially biased logged off-policy dataset $\mathcal{D}^n = \left(\boldsymbol{X}_j, A_j, R_j^{\text{obs}}\right)_{j=1}^n$ by first drawing an action $A_j \sim p_0(\cdot|\boldsymbol{X}_j)$, then setting the observed reward as $R_j^{\text{obs}} \equiv R_j(A_j)$. (We use $\kappa = 0.25$ for pendigits and $\kappa = 0.5$ for satimage and optdigits.) This bandit generation process makes the learning very challenging as the generated off-policy dataset has less number of observed labels.

We randomly divide the datasets into 70% training and 30% testing sets. We also randomly sequester 30% of the training set as a validation set. We train all algorithms for various parameter sets on the training set, validate the hyper parameters on the validation set and test on the testing set. We evaluate our algorithm with $L_r = 2$ representation layers, and $L_p = 2$ policy layers with 50 hidden units for representation layers and 100 hidden units (sigmoid activation) with policy layers. We implemented/trained all algorithms in a Tensorflow environment using Adam Optimizer.

For jth instance in testing data, let h_g^* denote the optimized policy of algorithm g. Let $\mathcal{J}_{test}$ denote the instances in testing set and $N_{test} = |\mathcal{J}_{test}|$ denote number of instances in testing dataset. We define (absolute) accuracy of an algorithm g as

$$\text{Acc}(g) = \frac{1}{N_{test}} \sum_{j \in \mathcal{J}_{test}} \sum_{a \in \mathcal{A}} h_g^*(a|\boldsymbol{X}_j) R_j(a).$$

We select the parameters $\lambda_1^* \in [0.005, 0.1]$, $\lambda_2^* \in [0, 0.01]$ and $\lambda_3^* \in [0.0001, 0.1]$ that minimize the loss given in (8) estimated from the samples in the validation set. In the testing dataset, we use the full dataset to compute the accuracy of each algorithm.

In the next subsection, we describe the performance of each algorithm on the third publicly available datasets. In each case, we run 25 iterations, following the procedure described above; we report the average of the iterations with 95% confidence intervals.

7.2.2 Results

In order to present a tough challenge to our algorithm we assume that the true propensities are not known and so must be estimated. Table 3 describes the absolute accuracy of each algorithm on each dataset. As can be seen, our algorithm outperforms all the benchmarks in each dataset within 95% confidence levels.

We define loss with respect to the "perfect" algorithm that would predict accurately all of the time, so the *loss* of the algorithm g is $1 - \text{Acc}(g)$. We evaluate the improvement of our

Table 3 Absolute accuracy in the UCI experiment (with 95% CI)

Algorithm/dataset	Pendigits (%)	Satimage (%)	Optdigits (%)
PONN-B	**88.01 ± 1.52**	**79.22 ± 0.42**	**79.98 ± 0.62**
PONN	85.45 ± 0.85	77.90 ± 0.45	75.46 ± 0.57
POEM-B	71.32 ± 0.73	45.15 ± 2.05	62.14 ± 0.75
POEM	68.98 ± 1.54	41.76 ± 2.05	59.49 ± 1.53
POEM-L1	70.84 ± 0.75	45.93 ± 1.01	60.75 ± 0.83
MLP-L1	83.16 ± 0.51	65.95 ± 6.42	75.28 ± 0.83
LR-L1	80.84 ± 0.35	67.45 ± 4.28	77.07 ± 0.07
Logging	10.12 ± 0.04	16.55 ± 0.54	10.24 ± 0.08

Bold values indicate "better" performance

Table 4 Improvement scores in the UCI experiment

Algorithm/dataset	Pendigits (%)	Satimage (%)	Optdigits (%)
PONN	17.59	5.52	18.41
POEM-B	58.19	61.93	47.12
POEM	61.34	64.32	50.58
POEM-L1	58.88	61.56	48.99
MLP-L1	28.80	38.97	53.71
LR-L1	37.42	36.15	19.01
Logging	86.65	75.09	77.69

algorithm over each other algorithm as the ratio of the actual loss reduction to the possible loss reduction, expressed as a percentage:

$$\text{Improvement Score}(g) = \frac{\text{Acc(PONN}-\text{B)} - \text{Acc}(g)}{1 - \text{Acc}(g)}$$

The Improvement Score of each algorithm g with respect to our algorithm is presented in Table 4. Note that our algorithm achieves significant Improvement Scores in all three datasets.

7.3 Chemotherapy regimens for breast cancer patients

In this subsection, we apply our algorithm to the choice of recommendations of chemotherapy regimen for breast cancer patients. We evaluate our algorithm on a dataset of 10,000 records of breast cancer patients participating in the National Surgical Adjuvant Breast and Bowel Project (NSABP) by Yoon et al. (2017). Each instance consists of the following information about the patient: age, menopausal, race, estrogen receptor, progesterone receptor, human epidermal growth factor receptor 2 (HER2NEU), tumor stage, tumor grade, Positive Axillary Lymph Node Count(PLNC), WHO score, surgery type, Prior Chemotherapy, prior radiotherapy and histology. The treatment is a choice among six chemotherapy regimes AC, ACT, AT, CAF, CEF, CMF. The outcomes for these regimens were derived based on 32 references from PubMed Clinical Queries. The rewards for these regimens were derived based on 32 references from PubMed Clinical Queries; this is a medically accepted procedure. The details are given in Yoon et al. (2017).

Using these derived rewards, we construct a dataset. In this dataset, an instance is described by a triple $(\boldsymbol{X}, A, R)$, where $\boldsymbol{X}$ is the 15-dimensional feature vector encoding the information about the particular patient, A is a chemotherapy regime, and R is the reward (survival/non-survival) for that chemotherapy regime for that patient. In the dataset, A is a chemotherapy regime generated in the same way as in the first experiment (with $\kappa = 0.25$) and R is the reward derived by Yoon et al. (2017).[4]

As in the previous experiment, in comparing algorithms, we consider absolute accuracy and the improvement score. In this context, we define the absolute accuracy of an algorithm g as the probability that its recommendation matches the chemotherapy regimen with the highest reward (according to best medical practice); i.e.

$$Acc(g) = \frac{1}{N_{test}} \sum_{j \in \mathcal{J}_{test}} \sum_{a \in \mathcal{A}} h_g^*(a|\boldsymbol{X}_j)\mathbb{I}(a = A_j^*)$$

[4] Unfortunately, our dataset does not record which chemotherapy regime was actually chosen for each patient.

Table 5 Performance in the breast cancer experiment

Metric	Accuracy (%)	Improvement (%)
PONN-B	**74.12 ± 1.25**	–
PONN	62.81 ± 1.85	30.41
POEM-B	55.39 ± 0.36	41.98
POEM	52.78 ± 0.50	45.19
POEM-L1	52.72 ± 0.55	45.26
MLP-L1	61.47 ± 0.50	32.00
LR-L1	51.96 ± 0.43	46.12
Logging	18.20 + 1.30	68.36

Bold values indicate "better" performance

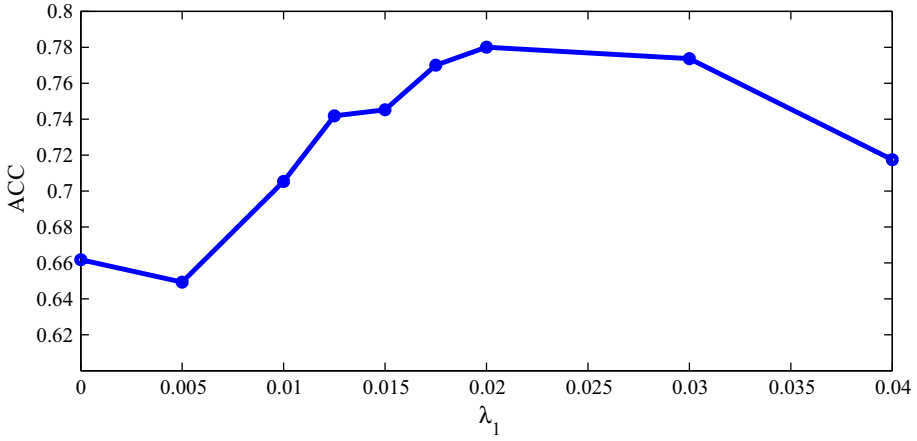

Fig. 2 Effect of the hyperparameter on the accuracy of our algorithm

As before, we define the Improvement Score with respect to relative loss.

Table 5 describes absolute accuracy and the Improvement Scores of the our algorithm. Our algorithm achieves significant Improvement Scores with respect to all benchmarks. There are two main reasons for these improvements. The first is that using Step B (feature selection) reduces over-fitting; this can be seen by the improvement of PONN-B over PONN and by the fact that PONN-B improves more over POEM (which does not use Step B) than over POEM-B (which does use feature selection). The second is that PONN-B allows for non-linear policies, which reduces model misspecification.

Note that our action-dependent relevance discovery is also important for interpretability. The selected relevant features given by our algorithm with $\lambda_1 = 0.03$ is as follows: age, tumor stage, tumor grade for AC treatment action, age, tumor grade, lymph node status for ACT treatment action, menopausal status and surgery type for CAF treatment action, age and estrogen receptor for CEF treatment action and estrogen receptor and progesterone receptor for CMF treatment action.

Figure 2 shows the accuracy of our algorithm for different choices of the hyper parameter λ_1. As expected—and seen in Fig. 2—if λ_1 is too small then there is overfitting; if it is too large then a lot of relevant features are discarded. We have chosen the value of λ_1 that maximizes accuracy.

8 Conclusion

This paper introduces a new approach and algorithm for the construction of effective policies when the dataset is biased and does not contain counterfactual information. The heart of our method is the ability to identify a small number of (most) relevant features—despite the bias and missing counterfactuals. When tested on a wide variety of data, the algorithm we introduce achieves significant improvement over state-of-the-art methods.

Acknowledgements This research was funded by Grants from NSF ECCS 1462245 and NSF IIP1533983.

Appendix

Here we collect the proofs of Theorems 1 and 2. It is convenient to begin by recording some technical lemmas; the first two are in the literature; we give proofs for the other two.

Lemma 1 (Theorem 1, Audibert et al. (2009)) *Let $X_1, X_2, \ldots, X_n$ be i.i.d. random variables taking their values in $[0, b]$. Let $\mu = \mathbb{E}[X_1]$ be their common expected value. Consider the empirical sample mean $\bar{X}_n$ and variance V_n defined respectively by*

$$\bar{X}_n = \frac{\sum_{i=1}^n X_i}{n} \text{ and } V_n = \frac{\sum_{i=1}^n (X_i - \bar{X}_n)^2}{n}. \tag{10}$$

Then, for any $n \in \mathbb{N}$ and $\delta \in (0, 1)$, with probability at least $1 - \delta$,

$$|\bar{X}_n - \mu| \leq \sqrt{\frac{2V_n \log 3/\delta}{n}} + \frac{3b \log 3/\delta}{n}. \tag{11}$$

For two probability distributions $\boldsymbol{P}$ and $\boldsymbol{Q}$ on a finite set $\mathcal{A} = \{1, 2, \ldots, a\}$, let

$$\|\boldsymbol{P} - \boldsymbol{Q}\|_1 = \sum_{i=1}^{a} |\boldsymbol{P}(i) - \boldsymbol{Q}(i)| \tag{12}$$

denote the L_1 distance between $\boldsymbol{P}$ and $\boldsymbol{Q}$.

Lemma 2 (Weissman et al. 2003) *Let $\mathcal{A} = \{1, 2, \ldots, a\}$. Fix a probability distribution $\boldsymbol{P}$ on $\mathcal{A}$ and draw n independent samples $\boldsymbol{X}^n = X_1, X_2, \ldots, X_n$ from $\mathcal{A}$ according to the distribution $\boldsymbol{P}$. Let $\widehat{\boldsymbol{P}}$ be the empirical distribution of $\boldsymbol{X}^n$. Then, for all $\epsilon > 0$,*

$$\Pr(\|\boldsymbol{P} - \widehat{\boldsymbol{P}}\|_1 \geq \epsilon) \leq (2^a - 2)e^{-\epsilon^2 n/2}. \tag{13}$$

The next two lemmas are auxiliary results used in the proof of Theorem 2.

Lemma 3 *Let $\boldsymbol{P}_0 = [p_0(a|\boldsymbol{x})]$ be the actual propensities and $\widehat{\boldsymbol{P}}_0 = [\widehat{p}_0(a|\boldsymbol{x})]$ be the estimated propensities. Assume that $\widehat{p}_0(a|\boldsymbol{x}) > 0$ for all $a, \boldsymbol{x}$. The bias of the truncated IS estimator with propensities $\widehat{\boldsymbol{P}}_0$ is:*

$$\begin{aligned}\text{bias}(\widehat{R}_m(a; \widehat{\boldsymbol{P}}_0)) = \sum_{j=1}^{n} \mathbb{E}\Bigg[\frac{\bar{r}(a, \boldsymbol{X}_j)}{n}\Bigg(\left(1 - \frac{p_{0,j}}{\widehat{p}_{0,j}}\right)\mathbb{I}\left(\widehat{p}_{0,j} \geq m^{-1}\right) \\ + \left(1 - p_{0,j}m\right)\mathbb{I}\left(\widehat{p}_{0,j} \leq m^{-1}\right)\Bigg)\Bigg].\end{aligned}$$

Proof of Lemma 3 The proof is similar to Joachims and Swaminathan (2016). We have

$$\begin{aligned}
\bar{r}(a) &= \frac{1}{n}\sum_{j=1}^{n}\mathbb{E}_{\boldsymbol{X}_j\sim\Pr(\mathcal{X})}\bar{r}(a,\boldsymbol{X}_j),\\
\mathbb{E}(\widehat{R}_m(a;\widehat{\boldsymbol{P}}_0)) &= \frac{1}{n}\sum_{j=1}^{n}\mathbb{E}_{(\boldsymbol{X}_j,A_j,R_j)\sim p_0}\left[\min\left(\frac{\mathbb{I}(A_j=a)}{\widehat{p}_0(A_j|\boldsymbol{X}_j)},m\right)R_j\right]\\
&= \frac{1}{n}\sum_{j=1}^{n}\mathbb{E}_{(\boldsymbol{X}_j,A_j)\sim p_0}\left[\min\left(\frac{\mathbb{I}(A_j=a)}{\widehat{p}_0(a|\boldsymbol{X}_j)},m\right)\bar{r}(a,\boldsymbol{X}_j)\right]\\
&= \sum_{j=1}^{n}\mathbb{E}_{\boldsymbol{X}_j\sim\Pr(\mathcal{X})}\left[\frac{\bar{r}(a,\boldsymbol{X}_j)}{n}\min\left(\frac{1}{\widehat{p}_0(a|\boldsymbol{X}_j)},m\right)p_0(a|\boldsymbol{X}_j)\right].
\end{aligned}$$

It follows that

$$\operatorname{bias}(\widehat{R}_m(a;\boldsymbol{P})) = \sum_{j=1}^{n}\mathbb{E}_{\boldsymbol{X}_j\sim\Pr(\mathcal{X})}\left[\frac{\bar{r}(a,\boldsymbol{X}_j)}{n}\left(1-\min\left(\frac{1}{\widehat{p}_0(a|\boldsymbol{X}_j)},m\right)p_0(a|\boldsymbol{X}_j)\right)\right]. \tag{14}$$

Dividing (14) into the case for which $\widehat{p}_0(a|\boldsymbol{X}_j)\geq m^{-1}$ and the case for which $\widehat{p}_0(a|\boldsymbol{X}_j)<m^{-1}$ and then combining the results yields the desired conclusion.

To state Lemma 4, we first define the expected relevance gain with truncated IPS reward using propensities $\widehat{\boldsymbol{P}}_0$ to be

$$g_m(a,i;\widehat{\boldsymbol{P}}_0) = \mathbb{E}\left[\left|\bar{r}_m(a,X_i;\widehat{\boldsymbol{P}}_0)-\bar{r}_m(a;\widehat{\boldsymbol{P}}_0)\right|\right]$$

where

$$\begin{aligned}
\bar{r}_m(a;\widehat{\boldsymbol{P}}_0) &= \mathbb{E}(\widehat{R}_m(a;\widehat{\boldsymbol{P}}_0))\\
&= \mathbb{E}_{(\boldsymbol{X},A,R)\sim p_0}\left[\min\left(\frac{\mathbb{I}(A=a)}{p_0(A|\boldsymbol{X})},m\right)R\right],\\
\bar{r}_m(a,x_i;\widehat{\boldsymbol{P}}_0) &= \mathbb{E}(\widehat{R}_m(a,x_i;\widehat{\boldsymbol{P}}_0))\\
&= \mathbb{E}_{(\boldsymbol{X},A,R)\sim p_0}\left[\min\left(\frac{\mathbb{I}(A=a)}{p_0(A|\boldsymbol{X})},m\right)R\middle|X_i=x_i\right].
\end{aligned}$$

□

Lemma 4 *We have:*

$$|g_m(a,i;\widehat{\boldsymbol{P}}_0)-g(a,i)| \leq B\left(\mathbb{E}\left[\left|\operatorname{bias}(\widehat{R}_m(a,X_i;\widehat{\boldsymbol{P}}_0))\right|\right]+\left|\operatorname{bias}(\widehat{R}_m(a;\widehat{\boldsymbol{P}}_0))\right|\right).$$

Proof of Lemma 4 This follows immediately by iterated expectations:

$$\begin{aligned}
&\left|\mathbb{E}\left(\ell\left(\mathbb{E}(\widehat{R}_m(a,X_i;\widehat{\boldsymbol{P}}_0))-\mathbb{E}(\widehat{R}_m(a;\widehat{\boldsymbol{P}}_0))\right)-\ell\left(\bar{r}(a,x_i)-\bar{r}(a)\right)\right)\right|\\
&\quad\leq\ B\mathbb{E}\left(\left|\mathbb{E}(\widehat{R}_m(a,X_i;\widehat{\boldsymbol{P}}_0))-\bar{r}(a,X_i)\right|\right)+\ B|\mathbb{E}(\widehat{R}_m(a;\widehat{\boldsymbol{P}}_0))-\bar{r}(a)|.
\end{aligned} \tag{15}$$

We now turn to the proofs of the theorems in the text. □

Proof of Theorem 1 Recall that the true relevance metric is $g(a,i) = \mathbb{E}\left[|\bar{r}(a,x_i) - \bar{r}(a)|\right] = \sum_{x_i \in \mathcal{X}_i} \Pr(X_i = x_i) l(\bar{r}(a,x_i) - \bar{r}(a))$. For any action $a \in \mathcal{A}$ and $x_i \in \mathcal{X}_i$, we can bound the error between the estimated relevance metric and the relevance metric as

$$
\begin{aligned}
|\widehat{G}(a,i;\boldsymbol{P}_0) - g(a,i)| = \Bigg| & \sum_{x_i \in \mathcal{X}_i} \frac{N(x_i)}{n} \ell\left(\widehat{R}(a,x_i;\boldsymbol{P}_0) - \widehat{R}(a;\boldsymbol{P}_0)\right) \\
& - \sum_{x_i \in \mathcal{X}_i} \frac{N(x_i)}{n} \ell\left(\bar{r}(a,x_i) - \bar{r}(a)\right) \\
& + \sum_{x_i \in \mathcal{X}_i} \frac{N(x_i)}{n} \ell\left(\bar{r}(a,x_i) - \bar{r}(a)\right) \\
& - \sum_{x_i \in \mathcal{X}_i} \Pr(X_i = x_i) \ell\left(\bar{r}(a,x_i) - \bar{r}(a)\right) \Bigg| \\
\leq & \sum_{x_i \in \mathcal{X}_i} \frac{N(x_i)}{n} \left(\ell\left(\widehat{R}(a,x_i;\boldsymbol{P}_0) - \widehat{R}(a;\boldsymbol{P}_0)\right) - \ell\left(\bar{r}(a,x_i) - \bar{r}(a)\right)\right) \\
& + \sum_{x_i \in \mathcal{X}_i} \left(\frac{N(x_i)}{n} - \Pr(X_i = x_i)\right) \ell\left(\bar{r}(a,x_i) - \bar{r}(a)\right) \\
\leq & B \sum_{x_i \in \mathcal{X}_i} \frac{N(x_i)}{n} \left|\widehat{R}(a,x_i;\boldsymbol{P}_0) - \bar{r}(a,x_i)\right| + B\left|\widehat{R}(a;\boldsymbol{P}_0) - \bar{r}(a)\right| \\
& + \sum_{x_i \in \mathcal{X}_i} \left|\frac{N(x_i)}{n} - \Pr(X_i = x_i)\right|.
\end{aligned}
$$

We bound each term separately. Applying Lemma 2, we see that with probability at least $1-\delta$, we have

$$
\begin{aligned}
\sum_{x_i \in \mathcal{X}_i} \left|\Pr(X_i = x_i) - \frac{N(x_i)}{n}\right| &\leq \sqrt{\frac{2\ln 2^{b_i}/\delta}{n}} \\
&= \sqrt{\frac{2\left(b_i \ln 2 + \ln 1/\delta\right)}{n}}. \qquad (16)
\end{aligned}
$$

Using Lemma 1 we see that, with probability at least $1-\delta$, we have

$$
\begin{aligned}
& \sum_{x_i \in \mathcal{X}_i} \frac{N(a,x_i)}{n} \left|\widehat{R}(a,x_i;\boldsymbol{P}_0) - \bar{r}(a,x_i)\right| \\
& \leq \sum_{x_i \in \mathcal{X}_i} \frac{N(a,x_i)}{n} \left(\sqrt{\frac{2V_n(a,x_i;\boldsymbol{P}_0)\ln 3/\delta}{N(a,x_i)}} + \frac{3M\ln 3/\delta}{N(a,x_i)}\right) \\
& \leq \sqrt{\frac{2b_i V_n(a,x_i;\boldsymbol{P}_0)\ln 3/\delta}{n}} + \frac{3Mb_i \ln 3/\delta}{n}, \qquad (17)
\end{aligned}
$$

where the the second inequality follows from an application of Jensen's inequality. Similarly, using Lemma 1, we see that with probability at least $1-\delta$, we have

$$\left|\widehat{R}(a;\boldsymbol{P}_0)-\bar{r}(a)\right| \leq \sqrt{\frac{2V_n(a;\boldsymbol{P}_0)\ln 3/\delta}{n}}+\frac{3M\ln 3/\delta}{n}. \tag{18}$$

The desired result now follows by combining (16, 17 and 18). □

Proof of Theorem 2 Let

$$\tilde{g}_m(a,i)=\sum_{c_i\in\mathcal{C}_{i,n}}\Pr(X_i\in c_i)\ell(\bar{r}_m(a,c_i)-\bar{r}_m(a)).$$

Then, we can decompose the error into

$$\begin{aligned}|\widehat{G}_m(a,i;\widehat{\boldsymbol{P}}_0)-g(a,i)| &\leq |\widehat{G}_m(a,i;\widehat{\boldsymbol{P}}_0)-g_m(a,i;\widehat{\boldsymbol{P}}_0)|+|g_m(a,i;\widehat{\boldsymbol{P}}_0)-g(a,i)|\\ &\leq |\widehat{G}_m(a,i;\widehat{\boldsymbol{P}}_0)-\tilde{g}_m(a,i;\widehat{\boldsymbol{P}}_0)|\\ &\quad+|\tilde{g}_m(a,i;\widehat{\boldsymbol{P}}_0)-g_m(a,i;\widehat{\boldsymbol{P}}_0)|\\ &\quad+|g_m(a,i;\widehat{\boldsymbol{P}}_0)-g(a,i)|. \end{aligned}\tag{19}$$

The first term (19) can be bounded by Theorem 1 by setting $s_n=\lceil n^{1/3}\rceil\leq n^{1/3}+1$, i.e.,

$$\begin{aligned}|\widehat{G}_m(a,i;\widehat{\boldsymbol{P}}_0)-\tilde{g}_m(a,i;\widehat{\boldsymbol{P}}_0)| \leq\ & \frac{\sqrt{4B^2\ln 3/\delta}}{n^{1/3}}\left(\sqrt{\bar{V}_n(a,i;\widehat{\boldsymbol{P}}_0)}+\sqrt{V_n(a;\widehat{\boldsymbol{P}}_0)}\right)\\ &+\frac{4mB\ln 3/\delta+\sqrt{2\ln 1/\delta+\ln 2}}{n^{2/3}}.\end{aligned}$$

The third term in (19) is the bias of the estimation due to estimated propensity scores and truncation, i.e.,

$$|g_m(a,i;\widehat{\boldsymbol{P}}_0)-g(a,i)|\leq B\left(\mathbb{E}\left[\left|\text{bias}(\widehat{R}_m(a,X_i);\widehat{\boldsymbol{P}}_0)\right|\right]+\left|\text{bias}(\widehat{R}_m(a);\widehat{\boldsymbol{P}}_0)\right|\right).$$

We bound the second term in (19)

$$\begin{aligned}g_m(a,i;\widehat{\boldsymbol{P}}_0) &= \mathbb{E}\left[\ell(\bar{r}_m(a,X_i;\widehat{\boldsymbol{P}}_0)-\bar{r}_m(a;\widehat{\boldsymbol{P}}_0))\right]\\ &= \mathbb{E}\left[\ell(\bar{r}_m(a,X_i;\widehat{\boldsymbol{P}}_0)-\bar{r}_m(a,c_i;\widehat{\boldsymbol{P}}_0)+\bar{r}_m(a,c_i;\widehat{\boldsymbol{P}}_0)-\bar{r}_m(a;\widehat{\boldsymbol{P}}_0))\right]\\ &\leq \mathbb{E}\left[\ell\left(\frac{L}{n^{1/3}}+\bar{r}_m(a,c_i;\widehat{\boldsymbol{P}}_0)-\bar{r}_m(a;\widehat{\boldsymbol{P}}_0)\right)\right]\\ &\leq \frac{LB}{n^{1/3}}+\mathbb{E}\left[\ell(\bar{r}_m(a,c_i;\widehat{\boldsymbol{P}}_0)-\bar{r}_m(a;\widehat{\boldsymbol{P}}_0))\right].\end{aligned}$$

where the first inequality follows from Assumption 3 and the second inequality follows from smoothness assumption on the loss function $l(\cdot)$, i.e.,

$$l\left(\frac{L}{n^{1/3}}+\bar{r}_m(a,c_i;\widehat{\boldsymbol{P}}_0)-\bar{r}_m(a;\widehat{\boldsymbol{P}}_0)\right)-l\left(\bar{r}_m(a,c_i;\widehat{\boldsymbol{P}}_0)-\bar{r}_m(a;\widehat{\boldsymbol{P}}_0)\right)\leq\frac{LB}{n^{1/3}}.$$

□

References

Alaa, A.M., van der Schaar, M. (2017). Bayesian inference of individualized treatment effects using multi-task gaussian processes. arXiv preprint arXiv:1704.02801

Atan, O., Zame, W. R., & van der Schaar, M. (2018). Learning optimal policies from observational data. arXiv preprint arXiv:1802.08679

Athey, S., & Imbens, G. W. (2015). Recursive partitioning for heterogeneous causal effects. arXiv preprint arXiv:1504.01132.

Audibert, J. Y., Munos, R., & Szepesvári, C. (2009). Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, *410*(19), 1876–1902.

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

Bottou, L., Peters, J., Candela, J. Q., Charles, D. X., Chickering, M., Portugaly, E., et al. (2013). Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, *14*(1), 3207–3260.

Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern classification*. Hoboken: Wiley.

Dudík, M., Langford, J., & Li, L. (2011). Doubly robust policy evaluation and learning. In *International conference on machine learning (ICML)*.

Dy, J. G., & Brodley, C. E. (2004). Feature selection for unsupervised learning. *Journal of Machine Learning Research*, *5*, 845–889.

Hall, M. A. (1999). Correlation-based feature selection for machine learning. PhD thesis, The University of Waikato

He, X., Cai, D., & Niyogi, P. (2005). Laplacian score for feature selection. In *Advances in neural information processing systems* (pp. 507–514).

Hoiles, W., & van der Schaar, M. (2016). Bounded off-policy evaluation with missing data for course recommendation and curriculum design bounded off-policy evaluation with missing data for course recommendation and curriculum design. In *International conference on machine learning* (pp 1596–1604).

Ionides, E. L. (2008). Truncated importance sampling. *Journal of Computational and Graphical Statistics*, *17*(2), 295–311.

Jiang, N., & Li, L. (2016). Doubly robust off-policy evaluation for reinforcement learning. In *International conference on machine learning (ICML)*.

Joachims, T., Grotov, A., Swaminathan, A., & de Rijke, M. (2018). Deep learning with logged bandit feedback. In *International conference on learning representations (ICLR)*.

Joachims, T., & Swaminathan, A. (2016). Counterfactual evaluation and learning for search, recommendation and ad placement. In *International ACM SIGIR conference on research and development in information retrieval* (pp 1199–1201).

Johansson, F., Shalit, U., & Sontag, D. (2016). Learning representations for counterfactual inference. In *International conference on machine learning (ICML)*

Kira, K., & Rendell, L. A. (1992). A practical approach to feature selection. In *Proceedings of the ninth international workshop on Machine learning* (pp. 249–256).

Koller, D., & Sahami, M. (1996). Toward optimal feature selection. Stanford InfoLab.

Maurer, A., & Pontil, M. (2009). Empirical bernstein bounds and sample variance penalization. In *The 22nd conference on learning theory*.

Peng, H., Long, F., & Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(8), 1226–1238.

Prentice, R. (1976). Use of the logistic model in retrospective studies. *Biometrics*, *32*(3), 599–606.

Robnik-Šikonja, M., & Kononenko, I. (2003). Theoretical and empirical analysis of relieff and rrelieff. *Machine Learning*, *53*(1–2), 23–69.

Rosenbaum, P. R., & Rubin, D. B. (1983). The central role of the propensity score in observational studies for causal effects. *Biometrika*, *70*(1), 41–55.

Shalit, U., Johansson, F., & Sontag, D. (2016). Estimating individual treatment effect: Generalization bounds and algorithms. arXiv preprint arXiv:1606.03976

Slivkins, A. (2014). Contextual bandits with similarity information. *Journal of Machine Learning Research*, *15*(1), 2533–2568.

Song, L., Smola, A., Gretton, A., Bedo, J., & Borgwardt, K. (2012). Feature selection via dependence maximization. *Journal of Machine Learning Research*, *13*(May), 1393–1434.

Strehl, A., Langford, J., Li, L., & Kakade S. M. (2010). Learning from logged implicit exploration data. In *Advances in neural information processing systems* (pp. 2217–2225).

Swaminathan, A., & Joachims, T. (2015). Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research*, *16*, 1731–1755.

Swaminathan, A., & Joachims, T. (2015b). The self-normalized estimator for counterfactual learning. In *advances in neural information processing systems* (pp. 3231–3239).

Tang, J., Alelyani, S., & Liu, H. (2014). Feature selection for classification: A review. *Data Classification: Algorithms and Applications, 37.*

Tekin, C., & van der Schaar, M. (2014). Discovering, learning and exploiting relevance. In *Advances in neural information processing systems* (pp. 1233–1241).

Tian, L., Alizadeh, A., Gentles, A., & Tibshirani, R. (2012). A simple method for detecting interactions between a treatment and a large number of covariates. arXiv preprint arXiv:1212.2995

Wager, S., & Athey, S. (2015). Estimation and inference of heterogeneous treatment effects using random forests. arXiv preprint arXiv:1510.04342

Weissman, T., Ordentlich, E., Seroussi, G., Verdu, S., & Weinberger, M. J. (2003). Inequalities for the l1 deviation of the empirical distribution. Hewlett-Packard Labs, Tech Rep.

Weston, J., Elisseeff, A., Schölkopf, B., & Tipping, M. (2003). Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, *3*, 1439–1461.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(3–4), 229–256.

Xu, Z., King, I., Lyu, M. R. T., & Jin, R. (2010). Discriminative semi-supervised feature selection via manifold regularization. *IEEE Transactions on Neural Networks*, *21*(7), 1033–1047.

Yoon, J., Davtyan, C., & van der Schaar, M. (2017). Discovery and clinical decision support for personalized healthcare. *IEEE Journal of Biomedical and Health Informatics*, *21*(4), 1133–1145.

Yu, L., & Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. *International Conference on Machine Learning (ICML)*, *3*, 856–863.