Benchmarking the Direct Method for Off-Policy Bandit Evaluation

Young Jin Park
yjp228@nyu.edu
New York University

Soojin Kim
sk5291@nyu.edu
New York University

Eric He
eh1885@nyu.edu
New York University

Tejomay Gadgil
tg1906@nyu.edu
New York University

Abstract

We benchmark Direct Methods using a variety of models on commonly used bandit datasets and find Direct Methods are rather unreliable compared to other methods. Simple direct methods consistently performs inferior to importance-weighted methods while improved versions, such as importance-weighted and doubly robust direct methods performs much better; yet still shares its limitations with simple direct methods. By using both actual bandit dataset and synthetic bandit datasets based on actual multiclass classification datasets, we explored how inductive Direct Methods are to different flavors of covariate shifts.

Our code can be found at https://github.com/EricHe98/direct_method.

1 Introduction

We are concerned with the problem of off-policy evaluation in the contextual bandit setting. The contextual bandit setting is as follows: given a *context* $x \in \mathcal{X}$ drawn from a distribution $\Omega(x)$, one of finitely many possible *actions* $a \in \mathcal{A}$ is selected according to a policy $\pi(a|x)$, and the action results in a real-valued *reward* $r \in \mathbb{R}$ drawn from a distribution $\delta(r|a,x)$. For any given policy π, we can estimate the *value* $V(\pi)$ of the policy by its expected reward[1]:

$$V(\pi) = \mathbb{E}_{\pi,\Omega(x)}[\delta(r|a,x)] \tag{1}$$

$$= \int_{x\in\mathcal{X}} \int_{a\in\mathcal{A}} \int_{r\in\mathbb{R}} \delta(r|a,x)\pi(a|x)\Omega(x)\, dr\, da\, dx \tag{2}$$

The goal of off-policy evaluation is to estimate the value of a *target* policy π_1 given a set of n tuples (x_0, a_0, r_0), (x_1, a_1, r_1), ..., (x_n, a_n, r_n) drawn from a *logging* policy π_0. The primary challenge here is that there is a mismatch between the actions taken by the logging policy in the data and the actions that would've been taken by the target policy given the same context.

[1]Usually, we regard $\Omega(x)$ as implicit and drop it from our notation:

$$\mathbb{E}_{\pi}[\delta(r|a,x)] = \int_{a\in\mathcal{A}} \int_{r\in\mathbb{R}} \delta(r|a,x)\pi(a|x)\, dr\, da$$

1.1 An example of off-policy evaluation in contextual bandits

An illustrative example of the contextual bandit setting is a doctor recommending treatments for patients with heart problems. The characteristics of the patient forms the context: for example, their age, height, weight, sex, reported level of pain and other relevant factors. The doctor's recommendation is the action. For example, the doctor could pick one of three recommendations: exercise, drugs, or surgery. The outcome of the patient is the reward. In this example, we could set it to 1 if the patient was "cured" and 0 otherwise. The doctor's criteria for recommendation forms the policy; for example, a policy could be to recommend exercise 75% of the time to anyone below the age of 60, and drugs otherwise; for patients of age 60 and above, recommend drugs 60% of the time and surgery otherwise.

We would like to evaluate a new doctor, but without going through the laborious process of giving recommendations to hundreds of patients and waiting to discover the outcome. One way this can be done is by grading the new doctor's treatment policy (π_1) according to the historical outcomes of patients treated by another doctor (π_0). If the two doctors recommend the same treatment for a patient, then we know the historical outcome of that patient is an exact assessment of π_1 for that patient. The issue is that if the two doctors do not recommend the same treatment for a patient, it's not possible to determine exactly what the outcome of the new doctor's policy would've been, since the data only records the outcome of the treatment recommended by π_0.

1.2 Approaches to off-policy evaluation

We present the two basic approaches to off-policy evaluation: *importance weighting* and *the direct method*. These two approaches are usually combined in a method called *doubly robust estimation*.

We note a key assumption of both methods: the logging policy must have support over all actions taken by the target policy. Mathematically, this means $\pi_0(a|x) > 0$ whenever

$\pi_1(a|x) > 0$. Intuitively, a target policy's action can only be evaluated if there is some probability of seeing it under the logging policy; otherwise the reward would be entirely unknown because the logging policy has never explored it. In the doctor example, if the new doctor recommends heart surgery to a newborn infant when the original doctor never has, there is no way to determine the outcome of that treatment using the original doctor's historical data.

1.2.1 Importance weighting. Given our dataset (x_0, a_0, r_0), (x_1, a_1, r_1), ..., (x_n, a_n, r_n) with the actions drawn from a logging policy π_0, the sample mean $\sum_{i=1}^{n} r_i$ is an unbiased estimator of the logging policy's value $V(\pi_0)$. Thus, our empirical estimate of $V(\pi_0)$ can be written as

$$\hat{V}(\pi_0) = \frac{1}{n}\sum_{i=1}^{n} r_i \tag{3}$$

However, what we are interested in is an estimate of $V(\pi_1)$. Importance weighting corrects the action sampling bias from the logging policy using a change of distribution to the target policy. Each data point is weighted by the relative likelihood of being seen in the target policy vs. the logging policy. Mathematically, this can be written as

$$V(\pi_1) = \mathbb{E}_{\pi_1}[\delta(r|a,x)] \tag{4}$$

$$= \int_{a\in\mathcal{A}}\int_{r\in\mathbb{R}} \delta(r|a,x)\pi_1(a|x)\, dr\, da \tag{5}$$

$$= \int_{a\in\mathcal{A}}\int_{r\in\mathbb{R}} \delta(r|a,x)\pi_1(a|x)\frac{\pi_0(a|x)}{\pi_0(a|x)}\, dr\, da \tag{6}$$

$$= \mathbb{E}_{\pi_0}[\delta(r|a,x)\frac{\pi_1(a|x)}{\pi_0(a|x)}] \tag{7}$$

and the empirical estimate of $V(\pi_1)$ would be

$$\hat{V}_{IW}(\pi_1) = \frac{1}{n}\sum_{i=1}^{n} r_i \frac{\pi_1(a_i|x_i)}{\pi_0(a_i|x_i)} \tag{8}$$

Importance weighting is guaranteed to give an unbiased estimate of $V(\pi_1)$.

The importance weights $\frac{\pi_1(a|x)}{\pi_0(a|x)}$ can take on any positive value; large importance weights can introduce significant variance into the value estimates. Reducing the variance around the estimates given by the standard importance weighting method is an active area of research; one method is *self-normalization*, in which we normalize by the sum of importance weights instead of by the count of data points:

$$\hat{V}_{SN\_IW}(\pi_1) = \left(\sum_{j=1}^{n}\frac{\pi_1(a_j|x_j)}{\pi_0(a_j|x_j)}\right)\sum_{i=1}^{n} r_i\frac{\pi_1(a_i|x_i)}{\pi_0(a_i|x_i)} \tag{9}$$

1.2.2 Direct method. Another well-known way to estimate rewards in a contextual bandit setting is via the direct method ("DM") of fitting a model to the rewards given the context and action. This simply casts rewards prediction problem as a standard supervised regression problem which can be solved with any off-the-shelf model.

For example, suppose we implemented the direct method by training $|\mathcal{A}|$ separate linear regressions $f_{a_1}(x), ..., f_{a_{|\mathcal{A}|}(x)}$ with weights $\theta_{a_1}, ..., \theta_{a_{|\mathcal{A}|}(x)}$ for each action with the square loss function. Then the training objective for the kth linear regression could be written as

$$\arg\min_{\theta_k\in\Theta}\frac{1}{n}\sum_{i=1}^{n}(r_i - \theta_k^T x_i)^2\mathbb{1}[a_i = k] \tag{10}$$

where $\mathbb{1}[a_i = k]$ is the indicator function taking value 1 if the ith action is k and 0 otherwise.

One flaw with the training objective written as is is that the model is trained on the logging policy's action distribution, and will try to minimize rewards error according to that action distribution. To get a model to minimize rewards error according to the target policy's action distribution, we can incorporate importance weighting into the loss function:

$$\arg\min_{\theta_k\in\Theta}\frac{1}{n}\sum_{i=1}^{n}(r_i - \theta_k^T x_i)^2\mathbb{1}[a_i = k]\frac{\pi_1(a_k|x_i)}{\pi_0(a_k|x_i)} \tag{11}$$

Given a fitted rewards estimator $\hat{f}_\theta(x_i, a_i)$, we can obtain our desired estimate $\hat{V}(\pi_1)$ by summing the predicted rewards for each action, weighted by π_1, over the values of x in our dataset. Mathematically, this would be written as

$$\hat{V}_{DM}(\pi_1) = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{|\mathcal{A}|}\hat{f}_\theta(x_i, a_k)\pi_1(a_k|x_i) \tag{12}$$

If $\hat{f}_\theta$ is an unbiased estimator of $\delta(r|a,x)\forall a, x$, then $\hat{V}_{DM}(\pi_1)$ will also be unbiased. This is a fairly strong condition, however, as it implies we would have a perfect rewards estimator.

It is expected for a regression model trained on a policy π_0 to generate an unbiased estimate of $\hat{V}(\pi_0)$, since this only requires the average prediction to equal the average target value: $\frac{1}{n}\sum_{i=1}^{n}\hat{f}_\theta(x_i, a_i) \approx \frac{1}{n}\sum_{i=1}^{n} r_i$. At least for the training set, seeing this equality is a natural outcome when using the square loss to train a model.

1.2.3 Doubly robust estimation. Doubly robust estimation combines importance weighting and the direct method. There are many doubly robust estimators; an example of one is

$$\hat{V}_{DR}(\pi_1) = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{V}_{DM}^{x_i}(\pi_1) + \frac{\pi_1(a_i|x_i)}{\pi_0(a_i|x_i)}(r_i - \hat{f}_\theta(x_i, a_i))\right), \tag{13}$$

$$\hat{V}_{DM}^{x_i}(\pi_1) = \sum_{k=1}^{|\mathcal{A}|}\hat{f}_\theta(x_i, a_k)\pi_1(a_k|x_i) \tag{14}$$

The doubly robust estimator has some nice properties. The estimate will be strictly better than IW or DM individually, and has lower variance than IW when the DM estimate is correlated with the true reward. Most interest around DM methods can be attributed to their utility in doubly robust estimators.

1.3 Utility of the direct method

DM is historically presented as an inaccurate method for off-policy evaluation, generally only useful in comparison to other approaches or as part of a doubly robust estimator.

From [3], the foundational paper on doubly robust estimation:

> ... the IPS estimator is in practice less susceptible to problems with bias compared with the direct method. However, IPS typically has a much larger variance...

From [12], the paper presenting the POEM estimator:

> In principle, since the logs give us an incomplete view of the feedback for different predictions, one could first use regression to estimate a feedback oracle for unseen predictions, and then use any supervised learning algorithm using this feedback oracle. Such a two-stage approach is known to not generalize well.

However, such papers tend to use simple and high-bias learning methods such as linear regression when implementing DM. In principle, complex nonlinear models such as neural nets or tree ensembles can be extremely effective DM estimators if they are able to produce accurate estimates of $\delta(r|a, x)$. Indeed, in the past decade, such models have seen much greater adoption and performance improvements on supervised learning problems. Recent research such as [2] has suggested that more complex models can give better performance on contextual bandits problems as well.

This paper contributes an empirical benchmark of the utility of DM on a number of datasets, using not just standard linear regression but also more complex nonlinear models such as gradient boosting and neural networks.

2 Related Work

The contextual bandits setting has rich ties to reinforcement learning, causal inference, missing value imputation, and other problems which have some concept of distribution mismatch. [3] established bounds on the bias and variance of doubly robust value estimators in the contextual bandits setting and provided the framework for translating full feedback datasets into bandit datasets which we use for our experiments; [7] and [5] do the same in the causal inference and reinforcement learning settings, respectively.

Estimators have been designed specifically for rewards prediction in partial feedback settings. The offset tree algorithm introduced in [1] is used by [3] for constructing doubly robust estimators; it trains a series of binary classifiers to distinguish whether to pick one action over another. The Q-learning algorithm introduced in [15] is a dynamic programming algorithm for producing estimates of the expected future reward of a given action and state in a reinforcement learning setting. The Slates estimator from [13] is specialized for off-policy evaluation in ranking problems, where every permutation of a set of items can be an action. For this paper, however, we are interested in using standard off-the-shelf regression and classification models.

Methods which use complex models to perform rewards estimation have seen great success. [6] introduced BanditNet, a method to train deep neural networks on contextual bandit feedback using a counterfactual risk minimization objective. [9] presented the deep Q-learning[2] method in a reinforcement learning setting, in which a deep convolutional neural network is used to estimate the reward function in playing a game of Atari. [2] presents a theoretical decomposition of excess risk into approximation error, estimation error, and "bandit error", in which IW methods are susceptible to bandit error but DM is not.

3 Datasets and Methodology

Natural bandit datasets are difficult to come by; this paper benchmarks on only two true bandit datasets, "Zozo" and "Hotels". To supplement this, we use publicly available classification datasets ("Yeast", "Statlog (Shuttle)", "DryBean", "Letter"). Summary statistics of each dataset are given in Table 1; we describe each dataset class in more detail below.

3.1 Bandit Datasets

3.1.1 Hotels. We pulled historical ranking logs from Rocketmiles, an online hotel booking platform, generated during an AB test of two different ranking policies. In this setting, the goal is to be able to use a rewards estimator trained on the logs of one policy to accurately predict the value of the second policy; doing so would obviate the need for a live AB test in which the new ranking policy is exposed to real life users, and allow for higher-quality offline experimentation.

The map to a bandit framework is as follows:

1. Each data point is an item (hotel).
2. The context x comprises user, query, and item (hotel) information such as the hotel price, whether the user interacted with the hotel before, etc.
3. The action a is the item's ranking induced by the policy; unfortunately, the policies were deterministic (albeit based on certain data fields not included in the context), so constructing a propensity estimator would be difficult.

[2]We call the rewards function $\delta(a|x, y)$, but in reinforcement learning, the function determining the expected future rewards for an action is called the Q function.

Table 1. Dataset characteristics

Name	Type	Data Points	Feature Count
Yeast	Classification (10)	1484	8
Statlog (Shuttle)	Classification (9)	58000	9
DryBean	Classification (7)	13611	15
Letter	Classification (26)	20000	16
Zozo	Bandit	10000	80
Hotels	Bandit	1648481	29

4. The reward r is the level of interaction the user had with the item: a 0 if the item was not interacted with, a 1 if the hotel details page was opened, a 2 if a specific hotel room was selected, and a 3 if the hotel room was booked.

We retain only the top 3 items in each search request, as those are immediately visible to the user without any need to scroll down the webpage and provide good signal about each policy's preferences. We look only at search requests in which the user made a booking (though the booked result did not have to be one of the top 3 items).

Evaluation is as follows:

1. One of the two policies is selected to be the logging policy; the other is the target policy. The dataset D_0 corresponding to the logging policy is split into a train set D_0^{train} and test set D_0^{test}. Splits are done by randomly partitioning based on search request; each set contains all the search results for half the search requests.
2. A rewards estimator $f(x)$ is trained on D_0^{train}. We tried linear regression, gradient boosting regression, and a gradient-boosting based ranking model called LambdaMART as rewards estimators.
3. D_0^{test} and D_1, the dataset corresponding to the target policy, serve as test sets. We predict the value on the two test sets and compare with the empirical test set value; the test set D_0^{test} from the same policy is our assessment of the ability of $f(x)$ to perform on-policy evaluation, while the test set D_1 for the target policy is our assessment of the ability of $f(x)$ to perform off-policy evaluation.

A modeling and evaluation loop can be performed M times with different bootstrap samples each time to build statistical estimates around the mean and standard deviation of the value estimates. We choose $M = 10$ for each of the two datasets.

3.1.2 ZOZO [11].

ZOZO is an online fashion retailer that uses multi-armed bandits to recommend clothes to users.

Similar to Rocketmiles, this dataset is data from a real A/B test of two ranking policies: Bernoulli Thompson Sampling (BTS) and a fully randomized policy. This data lets us evaluate off-policy performance for various estimators by comparing the BTS estimate on the random logged data with the ground truth value from the actual BTS logs.

While the full dataset consists of 26 million rows, we opted to use a subsample of 10,000 observations due to time and computational constraints.

Lastly, dataset was evaluated using a corresponding Python library, provided by ZOZO, called obp (Open Bandit Pipeline).

3.2 Evaluating on Classification Datasets

Classification datasets can be interpreted as bandit datasets where the optimal action is the correct label. We use publicly available multi-class classification datasets from the UC Irvine machine learning repository to simulate off-policy evaluation; a short description of each multi-class dataset is as follows:

1. **Yeast** [10]: each row is a protein, the target variable is the localization site and the features are outputs of various expert systems characterizing the protein and DNA sequence properties.
2. **Statlog (Shuttle)**: each row is a space shuttle design, the target variable is the type of space shuttle and the features are physical performance measures of the design.
3. **Dry Bean** [8]: each row is a photograph of a bean, the target variable is the type of dry bean and the features are derived from the photograph.
4. **Letter Recognition** [4]: each row is a photograph of a handwritten letter, the target variable is the letter and the features are derived from the photograph.

3.2.1 From full feedback to bandit feedback.

We mirror the procedure of [3] in converting a classification dataset into a bandit dataset. Suppose the dataset has K classes. Then each data point in the classification dataset can be written as (x, l) where x is the feature vector and $l \in 1, ..., K$ is the label corresponding to the correct class of x.

This data point can be converted into a bandit data point by:

1. keeping x as is.
2. mapping the action set $\mathcal{A}$ to the K class labels $1, ..., K$.
3. selecting an action $a \in 1, ..., K$ according to an as-yet undetermined policy.

4. setting the reward r to be 1 if the selected action matches the correct class and 0 otherwise: $\delta(r_i|a_i, x_i) = \mathbb{1}[a_i = l_i]$.

This methodology gives us full freedom to choose the logging and target policies.

3.2.2 Setting the target policy. The target policy can be an arbitrary function $\pi_1(a|x)$. We take a similar approach to [3] and [14]. First, we split the dataset into training and testing subsets with 70% and 30% of the data, respectively. Then, we train a classification model $\nu(x)$ on the train set with full feedback labels, and use that model's predicted probabilities as the target policy. The model's classification accuracy on the test set is analogous to the target policy's value.

The choice of setting our target policy to the predictions of a classification model trained on the full feedback replicates the intuition that the target policy should ideally be an improvement over the logging policy. We use logistic regression and gradient boosting trees as target policy classifiers.

3.2.3 Setting the logging policy. For one set of choices of logging policy, we again follow [3], which is as follows: with probability ϵ we select the "correct" action with reward 1, and the remaining $1-\epsilon$ probability is uniformly distributed across the incorrect actions. We vary ϵ to take different values between 0.1 and 0.8. This choice is intended to artificially replicate the notion that a logging policy has an imperfect, but better-than-random idea of the correct action to take. We term these logging policies "ϵ-correct logging policies".

Given such logging policy, we transform the training set into a bandit dataset, and use the partial feedback of the bandit dataset to build value estimators.

3.2.4 Rewards estimation. For each classification dataset, we benchmark the following models for DM:

1. Linear regression
2. Random forest regression

In addition to directly training on the bandit dataset, we also train models which reweight the loss of each data point based on the importance weight. These DM estimators are benchmarked against standard IW and SN_IW estimators, and used in the construction of DR estimators.

4 Results

4.1 Hotels

The results of the experiments are shown in Tables 2 and 3. Table 2 shows the performance of the estimators when evaluated on the holdout test set of data from the policy they were trained on, i.e. their on-policy performance, while Table 3 shows the corresponding off-policy performance on the data of the policy they were not trained on.

The two policies are labeled "OP" (old policy) and "NP" (new policy). For each policy and model, we report the following values:

1. Value: the empirical average reward of the evaluated policy across our bootstrap sampling runs:

$$\frac{1}{M}\sum_{j=1}^{M}\left(\frac{1}{N}\sum_{i=1}^{N} r_{ij}\right)$$

where $r_i j$ is the reward of the ith data point in the jth bootstrap sample.
The true value estimates on the bootstrap are fairly stable; the old policy has true value 0.137336 and the new policy has true value around 0.186435.
2. Bias: the average difference between the empirical value and the model's predicted value:

$$\frac{1}{M}\sum_{j=1}^{M}\left(\frac{1}{N}\sum_{i=1}^{N} r_{ij} - \frac{1}{N}\sum_{i=1}^{N}\hat{f}_j(x_{ij})\right)$$

3. Std: the standard deviation of the biases over the bootstrap samples:

$$\sqrt{\frac{1}{M}\sum_{j=1}^{M}\left(\text{Bias}_j - \text{Bias}\right)^2}$$

where $\text{Bias}_j = \left(\frac{1}{N}\sum_{i=1}^{N} r_{ij} - \frac{1}{N}\sum_{i=1}^{N}\hat{f}_j(x_{ij})\right)$ is the bias on the jth bootstrap sample.
4. RMSE: the average root mean squared error between the model's prediction and the true reward:

$$\frac{1}{M}\sum_{j=1}^{M}\left(\sqrt{\frac{1}{N}\sum_{i=1}^{N}(r_{ij} - \hat{f}_j(x_{ij}))^2}\right)$$

Note that this is a measure of the model's ability to estimate an individual reward, while the bias measures the model's ability to predict the average reward.

We did not find that more complex models like gradient boosting and random forest were able to decrease the bias over a linear regression baseline; in fact, simple linear regression had the least bias and RMSE among all the evaluated methods. The ranking model performed especially poorly. This is a reversal from the company's own metrics, where more complex ranking models had the best off-policy optimization performance. It's possible that this difference could be due to the top-3 filter we implemented, which severely limits the amount of data the DM models have access to and tends to sample a subset of the "best" items.

All DM estimators were extremely good at producing unbiased on-policy estimates, with the exception of the ranking-based model; in fact the DM estimator bias tends to be about the same size as the standard deviation in on-policy estimation. On off-policy estimates, however, the bias was one or

two orders of magnitude larger. In absolute terms, the off-policy estimates had errors of about 10% away from the true value, which makes it still usable for offline value estimates, but not good enough to replace AB testing.

4.2 ZOZO

For the ZOZO data we tested off-policy performance for inverse propensity weights (IPW), self-normalized inverse propensity weights (SN), as well as direct method (DM) and doubly-robust (DR) for three machine learning algorithms: Logistic Regression (LR), Gradient Boosting (GB), and Random Forest (RF).

The results show that IPW and SN_IPW are the most consistent for off-policy estimation. The quality of the direct method depends heavily on the underlying algorithm — while Gradient Boosting outperforms IPW, Random Forest does much worse. Direct Method also seems to have a very low standard deviation (SD), indicating variance is reduced in exchange for added bias.

Doubly-robust methods sit in between direct methods and IPW — while they do not outperform IPW, they are comparable, and they are considerably less biased than the direct method (in exchange for more variance, of course).

Based on the ZOZO trial, it seems that direct methods *can* outperform the importance-weighted approach in some cases, but further work is required to ascertain what those cases are, and whether they can be identifed in advance.

4.3 UCI: ϵ-correct logging policies

4.3.1 Covariate Shift. As mentioned in Section 3.2.3, for ϵ-correct logging policies, we experiment with five different values of ϵ (0.1, 0.2, 0.4, 0.6, and 0.8) and report the RMSE of the different estimators in Figure 1. The higher the ϵ, the stronger weight is put on the correct labels. With this method, we can roughly control the degree of the "quantitative" amount of covariate shift between our logging policy and target policy. For example, with Yeast dataset, when ϵ is 0.1, our logging policy's true value was 0.184 while our target policy's (Logistic Regression) true value was 0.431, in which case we have a covariate shift between the two policies.

We acknowledge that by changing ϵ, we can only control the distance between the means of logging policy's value and the target policy's value but not the similarity between the shapes of logging and target policies' distribution ("qualitative" aspect of covariate shift). Hence, experimenting with different ϵ values would have limited capability for accurately portraying the different degrees of covariate shifts. Later, we elaborate more on these limitations from the results of our experiments.

In Table 5, we observed that RMSE of the "mean estimator" is mostly coming from bias which we considered as proxy for the "quantitative" degree of covariate shift.Later, we elaborate more on these limitations from the results of our experiments.

When ϵ is small (i.e 0.1, 0.2), we have covariate shift because the logging policy and target policy have different distribution; however, compared to the bias of "mean estimator", which we roughly consider as the degree of covariate shift, all the value estimators including direct methods showed somewhat decent performance. With smaller ϵ, the sampling distribution of the logging policy is similar to population distribution of the dataset. In such case, even though we have clear covariate shift situation, direct method estimators performances were somewhat comparable to importance-weighted methods.

On the other hand, as ϵ increases, we have covariate shift situations where the sampled data from the logging policies carry stronger bias towards correct labels. Then we start to see rather interesting interactions between the logging policy and the target policies.

In Figure 1, for the Bean dataset, the Logistic Regression target policy has true policy value of 0.569, while the Gradient Boost target policy has true policy value of 0.901. In Logistic Regression target policy, the performance of direct methods deteriorates quickly as we introduce more covariate shifts; yet, when ϵ is small, direct methods performs better than when ϵ is large while RMSE of the "mean estimator" is smaller. For Gradient Boost target policy, since it's true policy value is close to 1, the RMSE of "mean estimate" decreases as ϵ increases. However, unlike Logistic Regression, direct method does not perform well when ϵ is small. The RMSEs of the "mean estimator" is minimized when ϵ is 0.4 and 0.8 for Logistic Regression and Gradient Boost target policies relatively. In the case of Gradient Boost target policy, by the design of the logging policy, we know that the distribution of the logging policy becomes similar (in shape) to the target policy. However, we cannot assume the same for Logistic regression; the shape of the distributions of logging and target policy does not have enough "similarity" for direct method to perform well even though the RMSE of the "mean estimator" is low.

Hence, in our ϵ-correct logging policy, the usage of RMSE of the "mean estimator" as proxy for degree of covariate shift has its limitations. RMSE of the "mean estimator" could be served as "quantitative" but not "qualitative" degree of covariate shift. We observed that it would be difficult to control the "qualitative" degree of covariate shift since we do not have an appropriate numeric measure to estimate the "similarities" in shape of the two distributions.

4.3.2 DM vs IW vs Doubly Robust method. In general, simple direct method, without importance weights, displayed inferior performance compared to importance-weighted methods. Direct methods were more inductive to bias coming from covariate shift as we expected.

Table 2. Hotel Rewards Estimation - On Policy Evaluation

	Name	True Value (On-Policy)	Bias	Std	RMSE
0	OP - Boosting	0.137233	-0.000772	0.000707	0.165774
1	OP - Linear Regression	0.137395	-0.000399	0.001017	0.132199
2	OP - LambdaMART	0.137365	-0.298251	0.002409	0.555895
3	OP - Random Forest Regression	0.137157	0.000016	0.000849	0.143139
4	NP - Boosting	0.186631	-0.000270	0.000734	0.184320
5	NP - Linear Regression	0.186394	-0.000720	0.000991	0.146489
6	NP - LambdaMART	0.186658	-0.250004	0.003538	0.576260
7	NP - Random Forest	0.186704	0.000409	0.001256	0.158106

Table 3. Hotel Rewards Estimation - Off Policy Evaluation

	Name	True Value (Off-Policy)	Bias	Std	RMSE
0	OP - Boosting	0.186435	-0.009941	0.001531	0.212893
1	OP - Linear Regression	0.186435	0.002677	0.000688	0.172784
2	OP - LambdaMART	0.186435	-0.442453	0.004265	0.699071
3	OP - Random Forest Regression	0.186435	-0.006005	0.001467	0.191687
4	NP - Boosting	0.137336	-0.010458	0.001054	0.158688
5	NP - Linear Regression	0.137336	-0.012108	0.000564	0.116803
6	NP - LambdaMART	0.137336	-0.115359	0.004427	0.418053
7	NP - Random Forest	0.137336	-0.023485	0.000603	0.132441

Table 4. ZOZO Off-Policy Evaluation (True value = 0.0042)

	Name	Estimate	Bias	SD
0	IPW	0.00458	0.000385	0.00194
1	SN_IPW	0.00481	0.000606	0.00220
2	DM_LR	0.00348	-0.000721	0.00002
3	DR_LR	0.00471	0.000505	0.00212
4	DM_GB	0.00390	-0.000298	0.00005
5	DR_GB	0.00470	0.000500	0.00208
6	DM_RF	0.00612	0.001922	0.00012
7	DR_RF	0.00456	0.000360	0.00228

Between direct methods with linear regression and random forest regression, random forest regression generally performed better then linear regression when more severe covariate shift was present. However, as covariate shift was less, random forest regression started to show the tendency for over-fitting, resulting in introducing more bias.

For Doubly Robust method introduced by [3], both direct methods with doubly robustness performed far superior to simple direct methods and in many cases, they performed marginally better than self-normalized importance weighted estimator (SN-IW). However, we have observed that when the performance of direct method deteriorates, doubly robust estimators' performance deteriorates as well.

In our experiments, overall, self-normalized importance weighted estimator performed the best compared to all other estimators when covariate shift was present. Occasionally, doubly robust estimators and importance weighted direct method estimators were marginally better than SN-IW estimator, the benefit of the both methods seems less appealing given the simplicity of SN-IW methods. We have seen doubly robust methods could get better with more expressive model (i.e. Linear Regression vs Random Forest), however, searching for appropriate model for direct methods and training the model may not be ideal for the benefit we gain.

Table 5. RMSE and Bias of different estimators on Yeast dataset

(a) RMSE

Estimators	Mean	IW	SN-IW	DM_{lin}	DM_{lin-IW}	DR_{lin}	DM_{rf}	DR_{rf}
$\epsilon = 0.1$	0.230586	0.038746	0.060437	0.123741	0.101019	0.084015	0.122617	0.101043
$\epsilon = 0.2$	0.149218	0.040624	0.044120	0.180675	0.075421	0.067780	0.178689	0.101710
$\epsilon = 0.4$	0.032347	0.024389	0.033129	0.305396	0.090825	0.080538	0.301397	0.164905
$\epsilon = 0.6$	0.212140	0.013737	0.056179	0.399540	0.139565	0.112046	0.385640	0.254786
$\epsilon = 0.8$	0.385685	0.009861	0.052449	0.473719	0.208767	0.126289	0.460377	0.302798

(b) Bias

Estimators	Mean	IW	SN-IW	DM_{lin}	DM_{lin-IW}	DR_{lin}	DM_{rf}	DR_{rf}
$\epsilon = 0.1$	-0.229663	0.015607	0.045150	0.122580	0.098681	0.081033	0.117334	0.096599
$\epsilon = 0.2$	-0.147416	0.014169	-0.007792	0.174921	0.033013	-0.006100	0.172575	0.087534
$\epsilon = 0.4$	0.027865	0.002688	-0.006319	0.304954	0.079110	0.023651	0.301176	0.160002
$\epsilon = 0.6$	0.211685	0.010169	0.041495	0.399312	0.137711	0.099735	0.385472	0.253085
$\epsilon = 0.8$	0.385618	0.009006	0.022552	0.473705	0.202949	0.079016	0.460348	0.301043

5 Discussion

There was no universal message across our experiments. The performance of direct method was inconsistent.

In terms of model complexity, more complex estimators produced better estimates than simpler estimators on certain datasets or sampling schemes and worse performance on others. For example, simple linear regression outperformed on the Hotels dataset, while random forest regression was the best performer on the Shuttle dataset. It seems that the performance of a DM model heavily depends on its ability to fit (and not overfit) the conditional rewards distribution $\delta(r|a, x)$.

Relative to other off-policy evaluation methods, the direct method would also outperform sometimes and underperform other times - although largely it underperformed. The importance weighted estimators tended to have the least bias. Doubly robust estimators combining both DM and IW were always better than the DM alone, but were unable to consistently outperform IW and self-normalized IW.

The importance weighting methods, when they were available, performed much more consistently across the multiclass datasets. Self-normalizing the importance weights was especially effective at culling the variance of the estimates.

6 Conclusion

To get a broader understanding of the utility of the direct method, we benchmarked on a variety of datasets, while using importance weighting and related methods for comparison when applicable. We found that the performances of DM estimators are heavily mixed depending on the dataset, model type, and choices of logging/target policies; no particular model was particularly better than the others, and it was not possible to tell *a priori* whether DM would work well on any given dataset. This is not entirely surprising, since a perfect DM estimator can be trivially converted into a reward-optimizing policy.

Although our experiments suggested importance weighting was a consistent performer, we did not run any benchmarks on cases where the true importance weights were unknown and had to be estimated; this is an area of future work that could better serve as comparison between DM and IW approaches.

Due to infrastructure and time constraint problems, we could only test on a number of smaller datasets, with most datasets being several thousand rows and the maximum dataset size being 1.6m rows; it's possible that DM methods were simply not leveraged to their fullest because they did not have a large amount of data to learn the conditional rewards distribution. Extending the benchmark to truly large datasets is another promising avenue of future work.

References

[1] Alina Beygelzimer and John Langford. "The Offset Tree for Learning with Partial Labels". In: *CoRR* abs/0812.4044 (2008). arXiv: 0812.4044. URL: http://arxiv.org/abs/0812.4044.

[2] David Brandfonbrener et al. "Overfitting and Optimization in Offline Policy Learning". In: *CoRR* abs/2006.15368 (2020). arXiv: 2006.15368. URL: https://arxiv.org/abs/2006.15368.

[3] Miroslav Dudık, John Langford, and Lihong Li. "Doubly Robust Policy Evaluation and Learning". In: *CoRR* abs/1103.4601 (2011). arXiv: 1103.4601. URL: http://arxiv.org/abs/1103.4601.

[4] Peter W. Frey and David J. Slate. "Letter Recognition Using Holland-Style Adaptive Classifiers". In: *Machine Learning* 6 (1991), pp. 161–182.

[5] Nan Jiang and Lihong Li. "Doubly Robust Off-policy Evaluation for Reinforcement Learning". In: *CoRR* abs/1511.03722 (2015). arXiv: 1511.03722. URL: http://arxiv.org/abs/1511.03722.

[6] Thorsten Joachims, Adith Swaminathan, and Maarten de Rijke. "Deep Learning with Logged Bandit Feedback". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=SJaP_-xAb.

[7] Joseph D. Y. Kang and Joseph L. Schafer. "Demystifying Double Robustness: A Comparison of Alternative Strategies for Estimating a Population Mean from Incomplete Data". In: *Statistical Science* 22.4 (2007), pp. 523–539. DOI: 10.1214/07-STS227. URL: https://doi.org/10.1214/07-STS227.

[8] Murat Koklu and Ilker Ali Ozkan. "Multiclass classification of dry beans using computer vision and machine learning techniques". In: *Computers and Electronics in Agriculture* 174 (2020), p. 105507. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2020.105507. URL: https://www.sciencedirect.com/science/article/pii/S0168169919311573.

[9] Volodymyr Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: *CoRR* abs/1312.5602 (2013). arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

[10] Kenta Nakai and Minoru Kanehisa. "A knowledge base for predicting protein localization sites in eukaryotic cells". In: *Genomics* 14.4 (1992), pp. 897–911. ISSN: 0888-7543. DOI: https://doi.org/10.1016/S0888-7543(05)80111-9. URL: https://www.sciencedirect.com/science/article/pii/S0888754305801119.

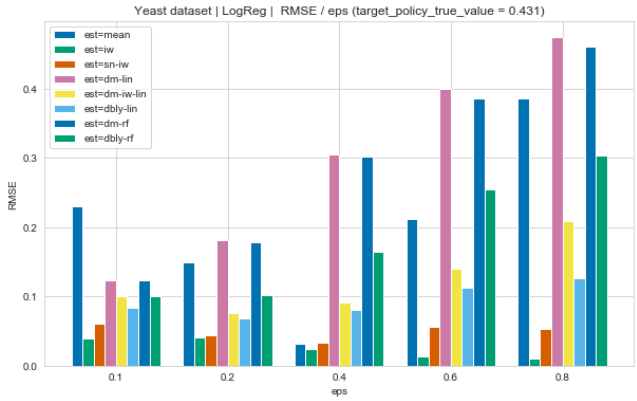
[11] Yuta Saito et al. *Open Bandit Dataset and Pipeline: Towards Realistic and Reproducible Off-Policy Evaluation*. 2021. arXiv: 2008.07146 [cs.LG].

[12] Adith Swaminathan and Thorsten Joachims. "Counterfactual Risk Minimization: Learning from Logged Bandit Feedback". In: *CoRR* abs/1502.02362 (2015). arXiv: 1502.02362. URL: http://arxiv.org/abs/1502.02362.

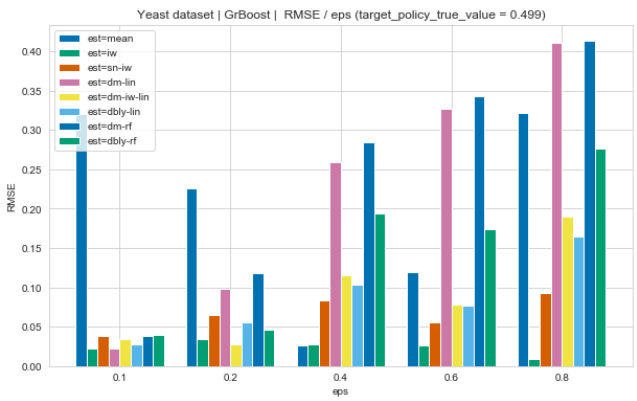
[13] Adith Swaminathan et al. *Off-policy evaluation for slate recommendation*. 2017. arXiv: 1605.04812 [cs.LG].

[14] Nikos Vlassis et al. "On the Design of Estimators for Bandit Off-Policy Evaluation". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 6468–6476. URL: http://proceedings.mlr.press/v97/vlassis19a.html.

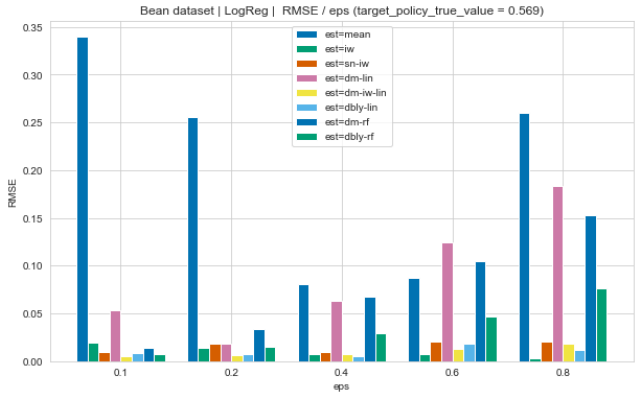
[15] Christopher J. C. H. Watkins and Peter Dayan. "Q-learning". In: *Machine Learning*. 1992, pp. 279–292.



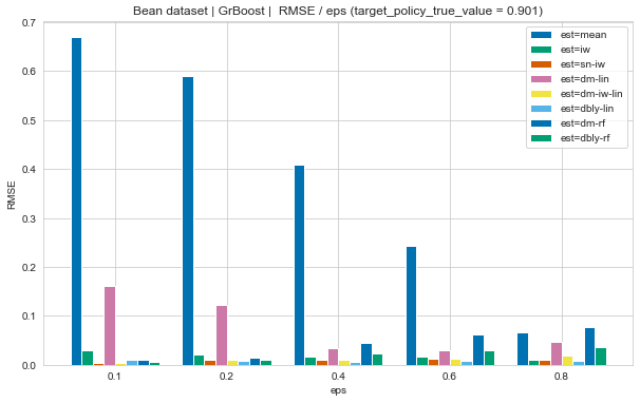
(a) Yeast data (n=1484) with LogReg as target policy



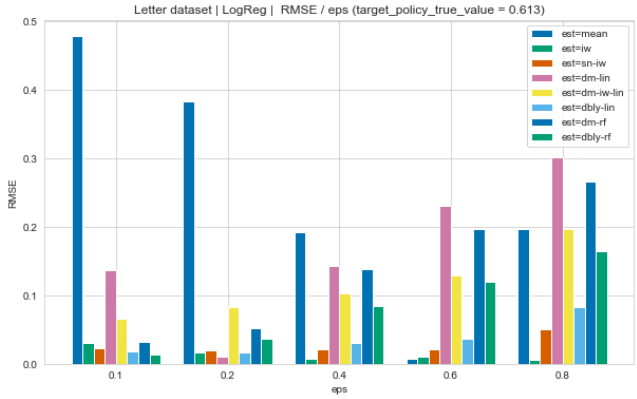
(b) Yeast data (n=1484) with GrBoost as target policy



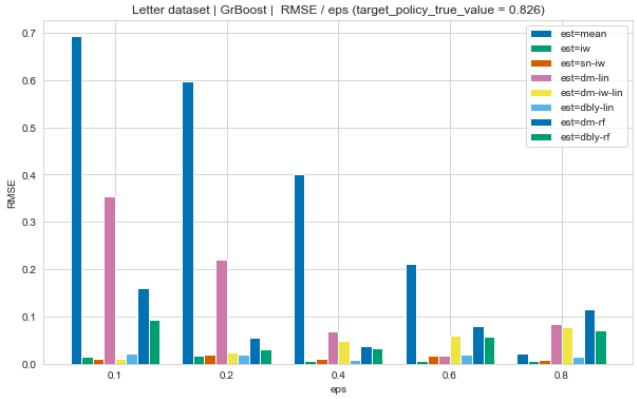
(c) DryBean data (n=13611) with LogReg as target policy



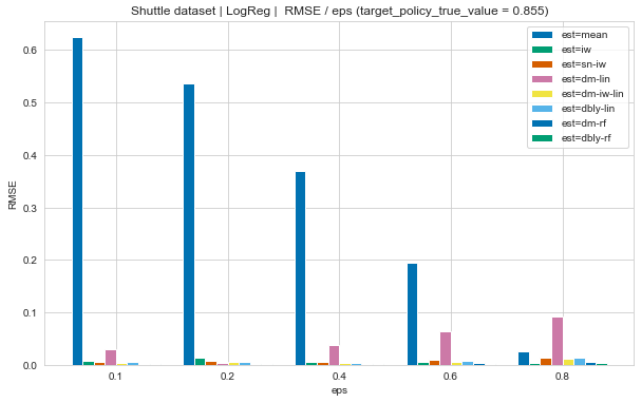
(d) DryBean data (n=13611) with GrBoost as target policy



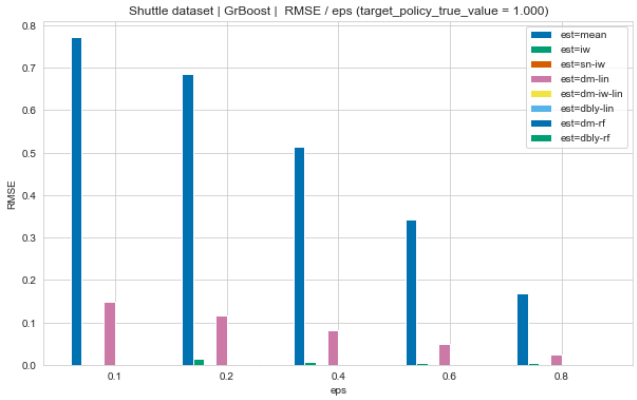
(e) Letter data (n=20000) with LogReg as target policy



(f) Letter data (n=20000) with GrBoost as target policy



(g) Shuttle data (n=58000) with LogReg as target policy



(h) Shuttle data (n=58000) with GrBoost as target policy

Figure 1. RMSE of estimators with different ϵ on UCI datasets