Contextual bandits and their Offline accomplices

prastogi

July 2020

1 Introduction

The following problems should be solved for deploying contextual bandits (decision-making agents) so that they converge faster, and accumulate lower regret when they begin operation.

1. **Offline Policy Evaluation** § 2
2. **Offline Policy Comparison** § 3
3. **Offline Policy Learning** § 4
4. Offline Bandit Evaluation
5. Offline Bandit hyper-parameter tuning and feature selection

1.1 The Problem, Assumptions and Notation

1. The actions are discrete, atomic, i.e. linearly enumerable, non-combinatorial.
2. Contexts are stochastic, **not** adversarial.
3. Rewards are non-stationary, but algorithms that assume stationary rewards are welcome. We'll use heuristics to deal with non-stationarity.
4. The action-policy, π^a, for a contextual bandit is contextual. I.e. the distribution depends on the customer features. π returns a single action. The corresponding symbol p returns a score-function or probability distribution.
5. There exists a dataset $\mathcal{D}$ of (feature-vector x, action a, reward y, action-distribution p). This dataset is gathered by a logging-policy π^l. The **logging-policy** is non-uniform, potentially contextual, and non-stationary.
6. $p_a(x, a)$ is the value of the action-policy conditional-distribution-function at (x, a). There is an unfortunate collision between action-policy and action. TODO: rename to behavior policy.

2 Offline Policy Evaluation (OPE)

Assumptions:

Action Policy	contextual, non-uniform, stationary
Logging Policy	contextual, non-uniform, non-stationary

The following is a representative list of OPE algorithms. For each approach I also mark the address of high quality implementations of these algorithms as well as references.

2.1 Direct Matching

Split $\mathcal{D}$ into two parts, $\mathcal{D}_t, \mathcal{D}_e$. Learn a predictive model $f \in \mathcal{F} :: x, a \to y$ using $\mathcal{D}_t$, and evaluate π_a on $\mathcal{D}_e$ using the estimate

$$\hat{V}_{\text{DM}}(\pi_a) = \frac{1}{|\mathcal{D}_e|} \sum_{i=1}^{|\mathcal{D}_e|} f(x_i, \pi_a(x_i)) = \frac{1}{|\mathcal{D}_e|} \sum_{i=1}^{|\mathcal{D}_e|} \sum_{a \in \mathcal{A}} f(x_i, a) p_a(x_i, a).$$

Due to the inconsistency of Direct-Matching (Section B) it should only be considered for use by itself on datasets collected by non-contextual, stationary logging policies. Any ML model for binary-classification or regression can be used to learn f.

2.1.1 Implementations

VW does not even implement DM for evaluation. They simply throw up an exception saying that evaluation with direct method is unbiased so it can't be used. But once we learn f then it's easy to evaluate the policy anyway. Also VW implemented doubly robust which I am sure can be tweaked to get DM instead.

Confidence Interval The simplest approach is to just CLT on $f(x_i, \pi_a(x_i))$. More sophisticated techniques can also use estimates of uncertainty in f. If the f is a calibrated classifier/regressor, e.g. a GP, or calibrated through isotonic regression, then we can estimate the confidence interval for each f, and do some smart-averaging.

2.2 IPS and SNIPS and their Balanced versions

$\hat{V}_{\text{IPS}}$ is estimated on the entire dataset $\mathcal{D}$.

$$\hat{V}_{\text{IPS}}(\pi_a) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} y_i \frac{p_a(x_i, a_i)}{p_{i,l}(a_i)} = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} y_i \frac{\mathbb{I}\{\pi_a(x_i) = a_i\}}{p_{i,l}(a_i)} \text{ worse implementation.}$$

$\hat{V}_{\text{SNIPS}}$ is also estimated on the entire dataset $\mathcal{D}$.

$$\hat{V}_{\text{SNIPS}}(\pi_a) = \frac{1}{\sum_{i=1}^{|\mathcal{D}|} \frac{p_a(x_i, a_i)}{p_{i,l}(a_i)}} \sum_{i=1}^{|\mathcal{D}|} y_i \frac{p_a(x_i, a_i)}{p_{i,l}(a_i)}.$$

SNIPS is much more robust to determinism in the logging policy. E.g. if the logging policy is deterministic, and it's wrong on 90% of the x and correct on 10% of x, the IPS estimator will say that the value of an action-policy that is correct on 100% of x will be 0.1. I.e. IPS will be be unnormalized.

These algorithms can be further implemented with a thresholding trick to minimize the impact of low-probability observations.

$$\hat{V}^{\tau}_{\text{IPS}}(\pi_a) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} y_i \frac{p_a(x_i, a_i)}{\max(\tau, p_{i,l}(a_i))} \quad \hat{V}^{\tau}_{\text{SNIPS}}(\pi_a) = \frac{1}{\sum_{i=1}^{|\mathcal{D}|} \frac{p_a(x_i, a_i)}{\max(\tau, p_{i,l}(a_i))}} \sum_{i=1}^{|\mathcal{D}|} y_i \frac{p_a(x_i, a_i)}{\max(\tau, p_{i,l}(a_i))}.$$

However note that the thresholding trick does not save us from determinism in the logging policy, and it adds another source of bias.

2.2.1 Balanced estimator and Balanced SNIPS

$$\hat{R}^{\text{BAL}}(\pi) = \frac{1}{n} \sum_{i=1}^{n} r_i \frac{\pi_a(a_i|x_i)}{\pi_{\text{avg}}(a_i|x_i)} \text{ where } \pi_{\text{avg}} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\theta_i}[(\mathbb{P}(a|x, \theta_i))].$$

Its SNIPS counterpart is balanced SNIPS.

2.2.2 Implementations

VW Estimators Repo

Confidence Interval Confidence interval calculation for IPS is presented in link. The first approach uses normality-of-averages (CLT). Basically treat $y_i \frac{p_a(x_i,a_i)}{p_{i,l}(a_i)}$ as iid random variables and use the standard-error and inverse-cdf of gaussian for estimating the confidence interval. Another method called clopper-pearson based on binomial distributions also has an incremental implementation. IMO the exact method is unimportant.

2.3 Doubly Robust evaluation

$$\hat{V}_{\text{DR}}^{\text{IPS}}(\pi_a) = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}(y_i - \hat{y}_i^f)\mathbb{I}\{\pi_a(x_i) = a_i\}/\max(p_{i,l}(a_i), \tau) + \hat{y}_i^f.$$

Here $\hat{y}_i^f = \sum_{a\in\mathcal{A}} f(x_i, a)p_a(x_i, a)$ for probabilistic action policies otherwise $\hat{y}_i^f = f(x_i, \pi_a(x_i))$.[1]

QUESTION What is the self-normalized version of doubly-robust? Do we let $w_i = p_a(x_i, a_i)/\max(p_{i,l}(a_i), \tau)$ and then

$$\hat{V}_{\text{DR}}^{\text{IPS}}(\pi_a) = \frac{1}{\sum w_i}\sum_{i=1}^{|\mathcal{D}|}(y_i - \hat{y}_i^f)w_i + \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\hat{y}_i^f.$$

or is it the self-normalized in a different manner?

2.3.1 [unimportant right now] Doubly robust off-policy evaluation with shrinkage

1907.09623.pdf, ICML

2.3.2 [unimportant right now] Triply Robust Off-Policy Evaluation

1911.05811v1.pdf, Use robust regression to train the DM component and then incorporate it into DR.

2.3.3 Implementation

See section 4.2.4 for the VW implementations of DR.[2]

2.4 [unimportant right now] The Empirical Likelihood Estimator

Karampatziakis, Langford, and Mineiro proposed an alternative for IPS/SNIPS in their paper "Empirical Likelihood for Contextual Bandits". This paper was presented at Neurips 2019 OptRL workshop and an impromptu session at COLT. Youtube.[3] The ELE can be seamlessly combined with DR or SWITCH (replacing their IPS part), and provides lower mean squared error.

Let $N = |\mathcal{D}|, w_i = \frac{p_a(x_i,a_i)}{p_{i,l}(a_i)}$. Assume w the importance sampling weights lie in a bounded interval.

$$\hat{V}_{\text{ELE}} = \sum_{i=1}^{N}\frac{1}{\beta^*(w_i) + N}w_i y_i.$$

Here β^* maximizes $\sum_{i=1}^{N}\log(\beta(w_n - 1) + N)$ subject to $\beta^*(w - 1) + N \geq 0$ for all w in the bounded interval mentioned above. They convincingly show that ELE is better than DR.[4] They also show how to estimate a confidence-interval in the paper in case $w_{\min}, w_{\max}$ are not observed. Also they note that by simply binning w and using the histogram representation of the problem the summation over N becomes a non-issue.

[1] Doubly robust gets its name because if either the reward predictor is unbiased or the logging-policy are accurate then the DR is unbiased, otherwise it has the lowest bias for reasonable logging policies.

[2] bandit_offline_doubly_robust.R provides an implementation of DR in the R-toolkit contextual.

[3] Sidenote: "A New Algorithm for Non-stationary Contextual Bandits: Efficient, Optimal, and Parameter-free" in COLT 2019by Chen et al. and "Adaptively Tracking the Best Bandit Arm with an Unknown Number of Distribution Changes" by Aeur et al. at COLT 2019 seemed interesting.

[4] There have been other improvements proposed for DR, such as MRDR but they are not really much better.

2.4.1 Implementation

Code is provided in pmineiro/elfcb and a readable version is given in VowpalWabbit.estimators.mle.py.

2.5 [unimportant right now]Cressie Read

Not important right now.

ELFCB Commit, ELFCB Ipython notebook, Cressie Read empirical likelihood is Cressie-Read with lambda 0 and the cressie-Read (lambda=-2) which is used because its computationally convenient.

3 Offline Policy Comparison

This is a simpler problem than offline-policy evaluation. because only the relative-difference between offline estimates of the policies needs to be correct as opposed to the absolute values. I'll just use a good policy evaluator for performing policy comparisons.

4 Offline Policy Learning

The naive-approach is guess-and-check where we offline optimize supervised-learning metrics such as AUC,MSE etc. and then run an online experiment to verify/quantify gain in CTR/satisfaction/revenue. This approach can result in negligible improvement in online metrics.

The other approach discussed in this section is a direct approach which measures the offline value-estimate of a policy and optimizes it.

4.1 Background

There are three different things/objects that we can learn from offline bandit feedback.

1. **Action predictor**. Directly learn a map from context to action that minimizes the counter-factual risk. E.g. POEM.
2. **Reward Expectation predictor**. Learn to estimate well the expected reward of each action, under each context, and then pick the action with best "expected reward". E.g. Logistic regression, Reweighted Logistic-Regression (either Non-linear least squares, or SGD), OLS, Reweighted-LS. The policy is implicitly coded as arg-max over the arms of the estimated expected reward per arm given context.
3. **Reward Distribution Predictor**, i.e. latent reward model, i.e. general reward moment predictor.[5] Instead of predicting just a single expectation **which forces the distribution to be a single bernoulli** predict a more complicated distribution, such as the linear-probit-model. methods such as BLIP. This gives rise to a randomized policy, also this learner can be used to warm-start thompson-sampling based online learners.

4.2 Policy learning

As mentioned in §**??** we are going to learn BLIP models for each content. The goal is to learn a blip model and then ultimately to prune the BLIP models based on the impact of the feature on the final prediction.

4.2.1 Online ADF for unweighted likelihood

The first method simply uses the BLIP algorithm for learning a stochastic reward estimator on the unweighted/vanilla likelihood.

If the features are sufficiently complex then there is no need for reweighting.

[5] Hybrid approaches are under research such as learning a policy using POEM and then compiling that policy into a distribution over rewards.

4.2.2 Laplace Approximation

Reweighted MAP.

4.2.3 Online ADF for weighted likelihood

Just optimize the expectation parameters and set variance heuristically.

The most basic approach uses the IPS estimator directly, to create a weighted-classification objective function by assigning to each example a weight of $1/p_{i,l}(x_i, a_i)$. This policy can be learnt on 70% of the offline data and evaluated on the remaning 30% of the offline data.

4.2.4 Implementations

Vowpal Wabbit provides a contextual bandits module in flag `--cb, --cb_explore_adf` which allows us to optimize predictor based on already collected data. The doubly-robust evaluator is used by default in VW. More info in vw cb tutorial and vw cb example and VW python wrapper for CB.

The doubly-robust estimator is implemented here.

`vw -d train.dat --cb 4 --cb_type ips|dr(default)|dm|mtr` .

5 More Off Policy Learning

The important thing to remember is that there are two types of off-policy learning problems. One where the policy-space is parameterized, and another where it isn't. When the policy space is not parameteric then the best we can do is off-policy evaluation and maybe put a GP (bayesian optimizer) on top, where the kernel induces a geometry on the parameters and how the performance might vary. In both of these situations we still need to be savvy about the off-policy evaluation metrics. One approach to off-policy learning is to reduce it to the problem of "Cost-sensitive classification", other ways are to reduce it to "Cost-sensitive regression" or "weighted regression". For "Cost sensitive classification" there are a few algorithms such as "DLM" and the "Filter-Tree reduction". The filter tree is a tree-of-tree. Each internal node is itself a binary classifier , so there is a small tree sitting inside each node.

5.1 The Basic Theory

http://alekhagarwal.net/bandits_and_rl/cb_intro.pdf http://alekhagarwal.net/bandits_and_rl/off_policy.pdf

The EXP4 algorithm needs to use O(policy) memory and computation and it experiences regret $O(\sqrt{KT \ln |\Pi|})$, where K is the number of actions that any policy can take. This means that if the number of possible actions is too large then a single step model of a structured prediction problem as a bandit problem will be screwed. Breaking down the algorithm's structured predictions into small steps is required to avoid this blowup. But that turns a structured prediction problem into a contextual bandit process.

IID Contextual Bandits (under full information) can be solved with a Greedy Algorithm.

1. Consider a bandit problem with full information (online learning) where the context and rewards are sampled IID. The greedy algorithm suffers at most $\sqrt{8T \log \frac{2NT}{\delta}}$.
2. Consider a contextual bandit problem with partial information where the context and rewards are sampled IID. The τ-greedy algorithm suffers at most $\sqrt{(TK)^{2/3}(\log \frac{N}{\delta})^{1/3}}$. But the main issue here is the fixed exploration that happens before we start exploiting.

http://www.cs.tau.ac.il/~mansour/papers/06jmlr.pdf This paper presented successive elimination.

1. Explore First
2. Epsilon-greedy

3. Successive Elimination
4. UCB based arm selection
5. Greedy and $\tau-$greedy

5.2 Off Policy Evaluation

See the bias and variance of IPS and DR in http://alekhagarwal.net/bandits_and_rl/off_policy.pdf.

5.3 Policy Optimizer for Exponential Models

We start with the paper, "Counterfactual Risk Minimization: Learning from Logged Bandit Feedback" at https://arxiv.org/pdf/1502.02362.pdf.

Minimizing the IPS has three issues

1. The minima of the IPS estimator is not invariant to additive transformations of the loss function and it gives degenerate results if the loss is not appropriately scaled. So we need to derive the optimal scaling for the loss function δ.
2. The IPS estimator can have unbounded variance. This problem can be fixed by "clipping" the importance sampling weights beyond some maximum value. The higher the upper threshold the lower the bias and higher the variance.
3. Finally the IPS estimator has different variance for different hypotheses. So just optimizing the IPS estimator without taking the variance into account is not useful.

So to counter all these problems they propose a better objective that explicitly takes the variance of the estimate into account. Specifically, if h maps an input x to a distribution over the output space $\mathcal{Y}$, or equivalently $h(x, y)$ assigns a probability value to each (x, y) pair, δ is the loss function between $[-1, 0]$ and the importance weights are clipped by the maximum value of M then the *counterfactual risk objective* is defined as

$$\hat{R}^M(h) + \lambda\sqrt{\mathrm{Var}_h(u)/n} \text{ Where} \tag{1}$$

$$\hat{R}^M(h) = \frac{1}{n}\sum_{i=1}^{n}\delta_i \min\left\{M, \frac{h(x_i, y_i)}{h_{\log}(x_i, y_i)}\right\} \tag{2}$$

$$\mathrm{Var}_h(u) = \frac{1}{n-1}\sum_{i=1}^{n}(u_i(h) - \bar{u}_i(h))^2 \tag{3}$$

$$u_i(h) = \delta(x_i, y_i)\min\left\{M, \frac{h(x_i, y_i)}{h_{\log}(x_i, y_i)}\right\} \tag{4}$$

However remember that the thing that we want to upper bound is $R(h) = \frac{1}{n}\sum_i \delta_i \frac{h(x_i,y_i}{h_{\log}(x_i,y_i)}$. Now in order to make $R^M(h) \geq R(h)$, δ must be smaller than zero. And in general if we have $\delta \in (a, b)$ then we can transport it to $[-1, 0]$ by the transformation $(\delta - b)/(b - a)$. Finally this paper gives a learning algorithm called POEM for conditional exponential distributional models.

The **POEM** method is basically a specific heuristic to optimize linear models via SGD wrt to the counterfactual objective. The software implementation is available at https://www.cs.cornell.edu/~adith/POEM/.

There were three important improvements made to this paper.

The first improvement was **SNIPS** and its optimizer called **Norm-POEM**. This was introduced in the paper "The Self-Normalized Estimator for Counterfactual Learning" by Swaminathan and Joachims. In this paper they propose to use the self-normalized IPS estimator

$$\hat{R}^{\text{SN}}(h) = \frac{\sum_i \delta_i h(x_i, y_i)/h_{\log}(x_i, y_i)}{\sum_i h(x_i, y_i)/h_{\log}(x_i, y_i)} \times \mathbb{E}[h(x, y)/h_{\log}(x, y)] \tag{5}$$

However $\mathbb{E}[h(x, y)/h_{\log}(x, y)] = 1$. But later on we actually need to use the thresholded versions of the importance weights, so we actually use

$$\hat{R}^{\text{SN}}(h) = \frac{\sum_i \delta_i \max\{M, h(x_i, y_i)/h_{\log}(x_i, y_i)\}}{\sum_i \max\{M, h(x_i, y_i)/h_{\log}(x_i, y_i)\}} \times \mathbb{E}[\max\{M, h(x, y)/h_{\log}(x, y)\}] \tag{6}$$

Then they use the delta method to estimate the variance of $\hat{R}^{\text{SN}}(h)$ to plug it into the CRM objective.

$$\hat{\text{Var}}(\hat{R}^{\text{SN}}(h)) = \frac{\sum_i (\delta_i - \hat{R}^{\text{SN}}(h))^2 (h(x_i, y_i)/p_i)^2}{(h(x_i, y_i)/p_i)^2} \tag{7}$$

The second improvement was in "Batch Learning from Bandit feedback through Bias corrected reward imputation" by Wang, Bai, Bhalla, and Joachims at https://www.cs.cornell.edu/people/tj/publications/wang_etal_19a.pdf. In this paper they deep-dived into the reward-regression approaches and addressed the issue that naive reward-estimators ("direct methods") typically have a large error in estimating the value of a policy. So let's say that $\delta(x, y) = \mathbb{E}[\text{reward} \mid x, y]$ the naive approach, IPS approach, and the BCRI approach train modified least-squares reward regression like this

$$\hat{L}_{\text{naive}}(\hat{\delta} \mid \text{data}) = \frac{1}{n} \sum_i (r_i - \hat{\delta}(x_i, y_i))^2 \tag{8}$$

$$\hat{L}_{\text{ips}}(\hat{\delta} \mid \text{data}) = \frac{1}{n} \sum_i \frac{1}{|\mathcal{Y}| h_{\log}(y_i \mid x_i)} (r_i - \hat{\delta}(x_i, y_i))^2 \tag{9}$$

$$\hat{L}_{\text{bcri}}(\hat{\delta} \mid h, \text{data}) = \frac{1}{n} \sum_i \frac{h(y_i \mid x_i)}{h_{\log}(y_i \mid x_i)} (r_i - \hat{\delta}(x_i, y_i))^2 \tag{10}$$

The paper shows that $\hat{L}_{\text{bcri}}(\hat{\delta} \mid h, \text{data})$ is an upper bound of the MSE of the policy value estimator. I.e. if the true empirical direct matching estimate of a policy is

$$R_{\text{DM}}(h \mid \delta, \text{dataset}) = \frac{1}{n} \sum_i \sum_{y \in \mathcal{Y}} h(y_i \mid x_i) \delta(x_i, y_i) \tag{11}$$

and the approximate empirical direct matching estimate of policy value is

$$\hat{R}_{\text{DM}}(h \mid \delta, \text{dataset}) = \frac{1}{n} \sum_i \sum_{y \in \mathcal{Y}} h(y_i \mid x_i) \hat{\delta}(x_i, y_i) \tag{12}$$

then $\hat{L}_{\text{bcri}}(\hat{\delta} \mid h, \text{data})$ upper-bounds the MSE $E[R(h) - \hat{R}(h)]$ where the error is introduced due to the approximation of δ using $\hat{\delta}$. Finally they propose to minimize $\hat{L}_{\text{bcri}}$ by jointly optimizing $\hat{\delta}$ and h as follows

$$h* = \arg\max_h [\hat{R}_{\text{DM}}] \; s.t. \; \hat{\delta} = \arg\min [\hat{L}_{\text{BCRI}}] \tag{13}$$

The third improvement was in "Doubly robust off-policy evaluation with shrinkage". This paper explores way of shrinking the importance weights in a systematic way to minimize upper bounds on the MSE of the doubly-robust policy-value estimators. Let $w(x, y) = \frac{h(y|x)}{h_{\log}(y|x)}$ then

$$\hat{V}_{\text{DR}}(h; \hat{\delta}) = \hat{V}_{\text{DM}}(h; \hat{\delta}) + \frac{1}{n} \sum_i w(x, y)(\delta_i - \hat{\delta}(x_i, y_i)) \tag{14}$$

$\hat{\delta}$ is assumed to have been trained on some weighted least-squares regression objective somehow. In this paper the focus is on modifications of the importance weight value w that is multiplied with the error term

in the DR estimator in equation 14. They substitute w with a quantity that minimizes the MSE of the DR estimator. ultimately they get a smoothed version of the IPS weighting as

$$\hat{w} = \frac{\lambda}{\lambda + w^2} w \qquad \textbf{Optimistic} \qquad (15)$$

$$\hat{w} = \min\{\lambda, w\} \qquad \textbf{Pessimistic} \qquad (16)$$

DETOUR

Warm starting Contextual Bandits: Robustly combining Supervised and Bandit Feedback
This paper does not specifically propose a method for Off-policy learning of contextual bandits, but rather it considers the question that how should "bandit feedback" and "supervised feedback" be mixed?

5.4 Learning from Extreme Bandit Feedback (POXM) and Deficient Support

The first paper by Lopez, Dhillon and Jordan studied the problem of batch learning from bandit feedback in the setting of extremely large action spaces.

The paper on "Off-Policy Bandits with Deficient Support" by Sachdeva, Su and Joachims at https://arxiv.org/pdf/2006.09438.pdf systematically studied mitigation strategies when the logging policy does not put any mass (or enough mass) on the actions proposed by the evaluated policy.

5.5 Model Inversion Networks

https://openreview.net/forum?id=SklsBJHKDS

The problem framework is that in a number of situations we are just solving optimization problems, where the function to optimize is learnt from data. For example, in off-policy optimization we have to learn a function from (context, action) to reward using the data, and then use that function (a.k.a model) to pick the best policy. Directly learning a model from (context, action) to reward for the purpose of then optimizing that function often doesn't work because the function becomes imperfect away from the training dataset. There are ways of handling this issue, like modeling the uncertainty in the function, or regularizing the argmax to stay close to the input data distribution (like a trust region for our learnt model). This paper presents a third approach which is to learn a "stochastic inverse map" as a model of a one-to-many relationship.

Then the paper presents three innovations

1. How the MIN should be constructed and trained? by minimizing – a reweighted version of – the expected divergence between conditional distributions. Instead of using a variational objective (ala. a KL Divergence) they chose to use a GAN (ala. a Jensen-Shannon objective). The reweighting is mentioned in § 3.3 which is motivated to be a product of a smoothed delta function and $N_y/(N_y + K)$ where N_y is just the number of points observed for a value of y.

2. How the argmax of the forward model should be computed using the MIN? compute the input to the MIN to be the values that maximize a forward model, but subject to lowest reconstruction loss, and highest probability to the encoding under the prior. This part is very much like an auto-encoder.

3.

A Empirical evaluations

Large-scale Validation of Counterfactual Learning Methods: A Test-Bed .

B Inconsistency of Direct Matching

Direct matching is an inconsistent estimator, i.e. it might return a wrong estimate of the value of a policy even with a single infinite dataset. There are two (more?) reasons for DM's inconsistency.

1. **Modeling Inconsistency** I.e. unrealizability. The feature-vector x and the model-family $\mathcal{F}$ do not contain the true map from $x, a \to y$. Some part of this is unavoidable, and other can be minimized by carefully solving the supervised learning problem.

2. **Sampling Inconsistency** Logging policies can distort the correlation between rewards and actions. For example, consider two scenarios.

   Scenario 1 Non-uniform, contextual, stationary logging policy can distort the correlation between rewards and actions.

If Male	Recommend Sport with 75% probability else Movie
If Female	Recommend Movie with 75% probability else Sports

   If the true click-through rate matrix is

	Male	Female
Sport	0.4	0.8
Movie	0.3	0.7

   . And Male-Female ration is 50:50, then our data will show that the overall CTR for Sports is 0.5, but the overall CTR for movies is 0.6. If for some reason this feature is not visible to the reward-predictor. The reward-predictor will smear the rewards incorrectly. Obviously **(contextual, non-stationary)** is an even harder case.

   Scenario 2 Non-uniform, non-contextual, non-stationary logging policies can cause distortions due to interactions with the non-stationarity of the context distribution. Consider the following policy.

On Day 1	Recommend Sport with 75% probability else Movie
On Day 2	Recommend Movie with 75% probability else Sports

   . If the true click-through rate matrix is

	Male	Female
Sport	0.4	0.8
Movie	0.3	0.7

   . And Male-Female ration is 50:50, but on day 1 there was a sale on men's goods, and on day 2 there was a sale on female's goods, so more men visited on day 1 and more women visited on day 2, then our data will show that the overall CTR for Sports is 0.5, but the overall CTR for movies is 0.6. The logging policy seems to be non-contextual but the non-stationarity in the environment has the same effect. Obviously **(contextual, non-stationary)** is an even harder case.

   Scenario 3: Non-uniform, non-contextual, stationary Let's say for every customer, on every day, Sports is recommending with 75% probability. Then this is an RCT with unequal treatment sizes, but still an RCT and the data will not have sampling inconsistency. In this situation direct-matching should work as long as the modeling inconsistency does not create problems.