Counterfactual Evaluation for Offline Contextual Bandits

David S. Rosenberg

NYU: CDS

October 13, 2021

Contents

Online vs offline in practice

Online learning

- We briefly discussed online supervised learning in the bandit module.
- Bandit methods inherently have an online learning component,
    - since policies should incorporate new information as soon as its received.

Main advantage of online learning

New information about the environment can be used immediately.

True online learning is pretty rare

- Consider a model recommending news stories on a webpage.
- Action: story recommendation.
- Reward: $\{0,1\}$ indicating a click.
- With multiple users, this does not cleanly fit into "rounds" of a bandit problem:
  - Time 0: User 1 gets news story recommendation.
  - Time 1: User 1 goes for coffee.
  - Time 2: User 2 arrives and wants a news story recommendation.
  - Time 10: User 1 returns and clicks on story.
  - Time $10+\varepsilon$: Model updates with new feedback and gives User 2 recommendation.
- In this example, a pure bandit is too rigid.

Online learning with delayed feedback

- We can allow feedback be delayed.
- Handle all requests with policy that we have.
- Update policy as we get feedback.
- The impact of this delayed feedback is investigated in [CL11, Table 1].
- Issue is compounded if multiple servers are running model in parallel.
  - How often to synchronize data across models?
  - This is related to the problem of asynchronous distributed model training.

Periodic batch retraining

- In practice, often sufficient and easier
  to periodically retrain the model.
  - e.g. daily, weekly, etc.
- In other words, use offline learning instead of online learning.
  - But refresh the model at regular intervals.

Online vs offline learning for safety

- In many business contexts,
  models require extensive testing before deployment.
- e.g. Assessing the model on a test set with multiple performance metrics.
- But online learning introduces another level of complexity.
- The model is changing on the fly.
- Can a particular sequence of feedback (possibly adversarial)
  - make the model worse?
- Seems safer to deploy a static model that we've tested.

Offline learning for bandits

- In the next couple modules,
  we'll discuss approaches to the bandit problem in the offline setting.
- This area goes by several names:
  - Counterfactual learning and evaluation
  - Offline bandit learning and evaluation
  - Batch learning with bandit feedback (BLBF)

Logged feedback

Stationary / Static Policies

- In general, the policy $\pi_t(\cdot \mid X_t, \mathcal{D}_t)$ at round t depends on the context X_t and the history $\mathcal{D}_t$.
- The dependence on the history $\mathcal{D}_t$ allows the policy to learn over time.
- In our study of offline bandits, we will deal with static or "stationary" policies.

Definition

A **stationary policy** or **static policy** maps from a context $x \in \mathcal{X}$ to a distribution over actions $\mathcal{A}$. We'll write $\pi(a \mid x)$ for the probability of taking action $a \in \mathcal{A}$ in context x under policy π.

Note that a stationary policy $\pi(a \mid x)$ has no dependency on t or $\mathcal{D}_t$.

- In the offline contextual bandit literature, stationary policies are usually just referred to as "policies", since they're the only ones typically under consideration.
- As usual, the terminology isn't essential. The notation tells the story very clearly. If the policy is written as $\pi(a \mid x)$, we know it's stationary.
- "But wait!" you say. What if somebody tries to get clever and put the history into x?
- Good question. This is basically what happens in the reinforcement learning (RL) setting. But remember that for the contextual bandit setting, we assumed that the contexts are generated i.i.d. That's the essential thing preventing us from putting the history $\mathcal{D}_t$ into the context X_t.
- Perhaps the key distinction between RL and the bandit setting is that consecutive contexts in RL are allowed to be dependent. This allows the context to accumulate information over time. Note that in RL, we speak of "states" rather than "contexts".

Stochastic contextual bandit with static policy

Stochastic k-armed contextual bandit with static policy π_0

1. Environment samples **context** and **reward vector** jointly, iid, for each round:

$$(X_1, R_1), \ldots, (X_n, R_n) \in \mathcal{X} \times \mathbb{R}^k \text{ i.i.d. from } P,$$

   where $R_i = (R_i(1), \ldots, R_i(k)) \in \mathbb{R}^k$.
2. For $i = 1, \ldots, n$,
    1. Our algorithm **selects action** $A_i \in \{1, \ldots, k\}$ according to $A_i \sim \pi_0(\cdot \mid X_i)$.
    2. Our algorithm **receives reward** $R_i(A_i)$.

- **Reminder**: $A_i \perp\!\!\!\perp R_i \mid X_i$.

Logged bandit feedback

- Suppose we run a static policy $\pi_0(a \mid x)$ on a contextual bandit for n rounds.
- The **logged bandit feedback** (i.e. the "history" or "observed data") is given by

$$(X_1, A_1, R_1(A_1)), \ldots, (X_n, A_n, R_n(A_n)),$$

  where $R_i(A_i) \in \mathbb{R}$ is the reward received in round i.
- The policy $\pi_0(a \mid x)$ is called the **logging policy**.
- Self-check: Are $(X_1, A_1, R_1(A_1)), \ldots, (X_n, A_n, R_n(A_n))$ i.i.d.?
  - Answer: Yes. (Because policy π_0 is static .)

- This is called **bandit** feedback specifically because in each round we only observe the reward corresponding to the action played.
- The analogous "full feedback" would be

  $$(X_1, A_1, R_1), \ldots, (X_n, A_n, R_n),$$

  where get the full reward vector in each round.
- We could solve the full feedback case by training a multiclass classifier to predict the action that has the highest reward in each round. This would be a straightforward supervised learning problem.
- With only bandit feedback, however, we're in the explore/exploit setting, since we won't know for sure whether or not a particular action is the best without trying the other actions.
- If the policy π_0 were not static, as in the online bandit setting, then logged bandit feedback from previous rounds, say $\mathcal{D}_n = ((X_1, A_1, R_1(A_1)), \ldots, (X_{n-1}, A_{n-1}, R_{n-1}(A_{n-1})))$ will generally give information about A_n, the action chosen in round n. In which case we would not have the independence claimed.

Off-policy learning and evaluation

- Suppose we have logged bandit feedback

$$\mathcal{D} = \left(\left(X_1, A_1, R_1(A_1) \right), \ldots, \left(X_n, A_n, R_n(A_n) \right) \right),$$

  from a contextual bandit with logging policy $\pi_0(a \mid x)$.
- We'll consider two problems for offline bandits:

Off-policy learning (counterfactual learning)

Use $\mathcal{D}$ from policy π_0 to **learn** a new policy π with better performance.

Off-policy evaluation (counterfactual evaluation)

Use $\mathcal{D}$ from policy π_0 to **estimate** the performance of a new policy π.

Off-policy bandits vs supervised learning

- This module will be about off-policy evaluation.
- Next module will be about off-policy learning.
- In supervised learning, the empirical risk on a test set is a great performance estimate.
- Optimizing the same empirical risk on a training set is the main approach to ML.
- In a bandit setting, getting an unbiased performance estimate is nontrivial,
  - and unbiased estimators can have high variance when π is very different from π_0.
- For off-policy learning, in addition to the overfitting concerns of supervised learning,
  - we have to deal with the different variances for our performance estimator.

The off-policy evaluation problem

The value function

Definition

The **value** of a static policy $\pi(x \mid a)$ in a contextual bandit problem is given by

$$V(\pi) = \mathbb{E}\left[R(A)\right],$$

where $(X, R) \sim P$ and $A \mid X \sim \pi(\cdot \mid X)$. The function $V(\cdot)$ is called the **value function**.

- The value function is the analogue of the risk function in supervised learning.
- The risk of f is the expected loss of f for a random (X, Y).
- The value of π is the expected reward for selecting A according to π.

Estimating value function from on-policy logs

- Suppose we have logged bandit feedback

$$\mathcal{D} = \left(\left(X_1, A_1, R_1(A_1) \right), \ldots, \left(X_n, A_n, R_n(A_n) \right) \right),$$

  from logging policy $\pi_0(a \mid x)$.
- Evaluating $V(\pi_0)$ is called **on-policy evaluation**,
  - since logged data is generated by the policy we want to evaluate.
- A natural estimator is

$$\hat{V}(\pi_0) = \frac{1}{n} \sum_{i=1}^{n} R_i(A_i).$$

- This is unbiased, since

$$\begin{aligned} \mathbb{E}\left[\hat{V}(\pi_0) \right] &= \mathbb{E}\left[R_i(A_i) \right] \\ &= V(\pi_0). \end{aligned}$$

Estimating value function from off-policy logs

- Suppose we want to estimate $V(\pi)$,
  where π is different from the logging policy π_0.
- This is called **off-policy evaluation**.
- We can write

$$V(\pi) \quad = \quad \mathbb{E}_{(X,R)\sim P, A|X\sim\pi(\cdot|X))} [R(A)] .$$

- We want to estimate the expectation of $R(A)$
  - when $A \mid X \sim \pi(\cdot \mid X)$, but
  - we have $A \mid X \sim \pi_0(\cdot \mid X)$ in our data $\mathcal{D}$.
- We have a distribution mismatch.
- What can we do in this situation?

Estimating value function from off-policy logs

- Just like the missing at random (MAR) setting, there are multiple methods for estimating $V(\pi)$.
- For each MAR estimator, there is an analogous policy value estimator:

MAR estimator	Off-policy value estimator
IPW mean	importance-weighted (IW) value estimator / "Model the bias"
self-normalized IPW mean	self-normalized IW value estimator [SJ15]
regression imputation	"Direct Method" / "Reward prediction"
augmented IPW	"doubly robust" estimator [DLL11]

Direct methods / "Model the world"

The obvious thing?

- We want to estimate

$$\begin{aligned} V(\pi) &= \mathbb{E}\left[R(A)\right] \\ &= \mathbb{E}_X\left[\mathbb{E}_{R,A}\left[R(A) \mid X\right]\right], \end{aligned}$$

  where $A \sim \pi(\cdot \mid X)$.
- We always have $(X, R) \sim P$, so won't mention that going forward.
- We have plenty of X's from the right distribution in our logs.
- So the outer expectation is easy to estimate using Monte Carlo.

If we knew the expected rewards...

- We want to calculate $V(\pi) = \mathbb{E}\left[\mathbb{E}\left[R(A) \mid X\right]\right]$.
- Let $r(x,a) = \mathbb{E}\left[R(A) \mid X = x, A = a\right]$.
- Then

$$
\begin{aligned}
\mathbb{E}\left[R(A) \mid X\right] &= \mathbb{E}\left[\mathbb{E}\left[R(A) \mid X, A\right] \mid X\right] \\
&= \mathbb{E}\left[r(X,A) \mid X\right]
\end{aligned}
$$

- Putting it together,

$$
\begin{aligned}
V(\pi) &= \mathbb{E}_X\left[\mathbb{E}_{A \sim \pi(\cdot \mid X)}\left[r(X,A) \mid X\right]\right] \\
&= \mathbb{E}_X\left[\sum_{a=1}^{k} r(X,a)\pi(a \mid X)\right].
\end{aligned}
$$

Estimate with expected rewards

- We have

$$V(\pi) \quad = \quad \mathbb{E}_X\left[\sum_{a=1}^{k} r(X,a)\pi(a \mid X)\right]$$

- Create a Monte Carlo style estimator using our sample of contexts:

$$\hat{V}(\pi) = \frac{1}{n}\sum_{i=1}^{n}\left[\sum_{a=1}^{k} r(X_i,a)\pi(a \mid X_i)\right].$$

- But we don't know $r(x,a)$ – what can we do?

Estimating the rewards

- Can we estimate $r(x,a) = \mathbb{E}[R(A) \mid X = x, A = a]$?
- (We discussed this in the contextual bandits module.)
- Our logs gives us data for this: $(X_i, A_i, R_i(A_i))_{i=1}^n$.
- This is a straightforward regression problem.
- Ok... there **is** a **covariate shift** here.
- We want to apply $r(x,a)$ when $A \sim \pi(\cdot \mid x)$, but we have samples from $A \sim \pi_0(\cdot \mid x)$.
  - In this context, it's sometimes called a **selection bias**.
- Once we have an estimate $\hat{r}(x,a)$, we can plug it in.

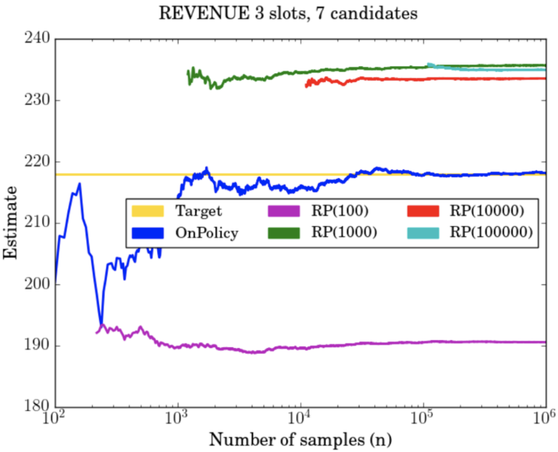
Definition

The **direct method** of offline policy evaluation is

$$\hat{V}_{\text{dm}}(\pi) = \frac{1}{n} \sum_{i=1}^{n} \sum_{a=1}^{k} \hat{r}(X_i, a) \pi(a \mid X_i).$$

- As we know from our study of regression imputation, importance weighting becomes important when we have model misspecification and response bias / covariate shift.
- HOWEVER, in practice, we often don't want to evaluate just one policy. We often want to evaluate policies in a hyperparameter tuning setting, where we may want to evaluate 10-20 different policies. To use importance weighting, we'd have to train a different $\hat{r}(x, a)$ for each policy we want to estimate. And perhaps we should, if 1) we have the time, and 2) we think we're in the model misspecification setting (i.e. the model we're fitting to the data doesn't have a good approximation to $\mathbb{E}[R(A) \mid X = x, A = a]$).
- For learning, the situation is even worse – as we traverse through policy space, the required importance weighting would be constantly changing. They actually try to use importance weighting to adjust accordingly in this scenario in [WBBJ19].

Experiment from Swaminathan and Joachims

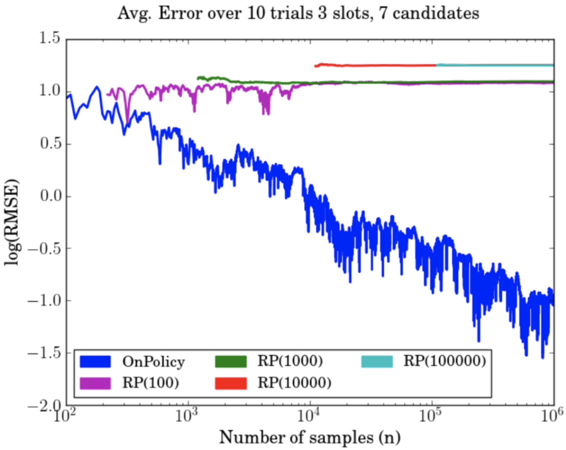


- RP = Reward prediction / direct method with various training set sizes

From https://www.cs.cornell.edu/~adith/CfactSIGIR2016/Evaluation1.pdf page 21.

- This was a simulated news recommender scenario. The reward function was a very complicated function of the context and the action.
- Because it's a simulation, we know the actual policy value – that's the "target" yellow line.
- We can also actually run the policy we want to evaluate against the simulator and get an on-policy value estimate. That's the blue line. We can see that it converges fairly well to the target.
- Each of the lines marked RP(X) is an instance of a direct method. For example, RP(1000), the green line, used the first 1000 points of logged data to train the reward model. Then the green line shows the value estimate as we use the rest of the data.
- The direct method (the RP(X) lines) don't bounce up and down much – that's low variance. But they're quite a bit off from the target. My best guess for the poor performance is that it's a combination of model misspecification and covariate shift. It would be interesting to dig in more to this simulation to identify what the issue is, including looking at the actual reward function. Are they using an hypothesis space that's very limited?

Experiment from Swaminathan and Joachims



- RP = Reward prediction / direct method with various training set sizes

From https://www.cs.cornell.edu/~adith/CfactSIGIR2016/Evaluation1.pdf page 21.

- Same thing on a log scale, averaged over 10 trials.

What's going on here?

- We've studied regression imputation before.
- We know we have problems when there's
  - model misspecification AND
  - sample bias
- Maybe we can do something about the sample bias with importance weighting?
- Maybe we can reduce model misspecification with a more complex model?
- Model misspecification is a fact of life with finite data.
  - We intentionally "misspecify" with a simpler hypothesis space so we don't overfit.

Importance-weighted value estimators

Our approach

- In the missing data setting, we get a biased sample.
- The complete case mean gives a biased estimate.
- IPW mean was one approach to get an unbiased estimate.
- IPW works by reweighting each observation appropriately.
- In the offline bandit setting, our observations are the rewards we receive
  - from running the logging policy π_0.
- Our observations are "biased" compared to the policy π we're evaluating.
- In this section, we'll also use a reweighting scheme (importance sampling)
  - to get an unbiased estimate of policy value.

Value function

- Write $p(x, r)$ for the joint PDF/PMF of X and R.
- Then we can rewrite the value of π as follows:

$$\begin{aligned} V(\pi) &= \mathbb{E}_{(X,R)\sim P, A\sim\pi}\left[R(A)\right] \text{ (definition)} \\ &= \mathbb{E}_{(X,R)\sim P, A\sim\pi_0}\left[\frac{p(X,R)\pi(A \mid X)}{p(X,R)\pi_0(A \mid X)} R(A)\right] \text{ (change of measure theorem)} \\ &= \mathbb{E}_{(X,R)\sim P, A\sim\pi_0}\left[\frac{\pi(A \mid X)}{\pi_0(A \mid X)} R(A)\right], \end{aligned}$$

  where the change of measure requires that $\pi_0(a \mid x) = 0 \implies \pi(a \mid x) = 0$.
- We can use logged data to make an unbiased Monte Carlo estimate of the last expression.

If you prefer not to assume a PDF or PMF for X and R, we can avoid it as follows: Let

$$
\begin{aligned}
g(x,r) &:= \mathbb{E}_{A\sim\pi(\cdot|x)}\left[R(A)\mid X=x, R=r\right] \\
&= \mathbb{E}_{A\sim\pi(\cdot|x)}\left[r(A)\right] \\
&= \mathbb{E}_{A\sim\pi_0(\cdot|x)}\left[r(A)\frac{\pi(A\mid x)}{\pi_0(A\mid x)}\right] \text{ (change of measure / importance sampling)}
\end{aligned}
$$

In the second equality, since we're conditioning on the reward vector $R=r$, we can just replace $R(A)$ by $r(A)$. Next, we've already fully specified the distribution of A in the subscript of $\mathbb{E}_{A\sim\pi(\cdot|x)}$, and nothing else in $r(A)$ is random, so we can drop the conditioning on $X=x$ and $R=r$. Then

$$
\begin{aligned}
V(\pi) &= \mathbb{E}_{(X,R)\sim P, A\sim\pi}\left[R(A)\right] \\
&= \mathbb{E}_{(X,R)\sim P}\left[\mathbb{E}_{A\sim\pi(\cdot|X)}\left[R(A)\mid X,R\right]\right] \\
&= \mathbb{E}_{(X,R)\sim P}\left[g(X,R)\right] \\
&= \mathbb{E}_{(X,R)\sim P}\left[\mathbb{E}_{A\sim\pi_0(\cdot|X)}\left[R(A)\frac{\pi(A\mid X)}{\pi_0(A\mid X)}\right]\right]
\end{aligned}
$$

Importance-weighted value estimator

Definition

The **importance-weighted (IW) value estimator** for policy π, based on logged bandit feedback $(X_1, A_1, R_1(A_1)), \ldots, (X_n, A_n, R_n(A_n))$ with actions chosen under static logging policy π_0 is

$$\hat{V}_{\text{iw}}(\pi) = \frac{1}{n} \sum_{i=1}^{n} R_i(A_i) \frac{\pi(A_i \mid X_i)}{\pi_0(A_i \mid X_i)}.$$

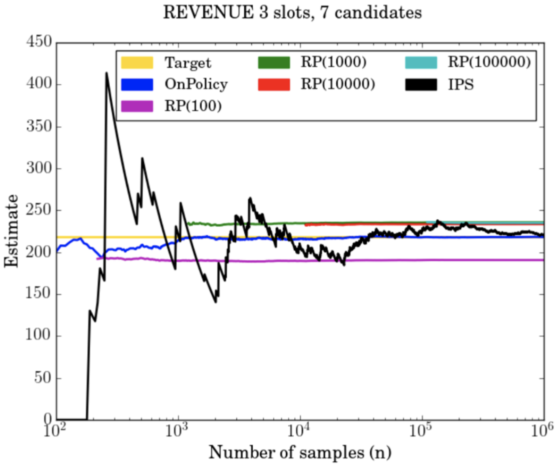
The **importance weights** are defined as $W_i = \pi(A_i \mid X_i)/\pi_0(A_i \mid X_i)$.

- The following follows immediately from the importance sampling formulation of $V(\pi)$:

Theorem (The IW estimator is unbiased.)

If $\pi_0(a \mid x) = 0 \implies \pi(a \mid x) = 0$ *for all* $(x, a) \in \mathcal{X} \times \mathcal{A}$, *then* $\mathbb{E}\left[\hat{V}_{ipw}(\pi)\right] = V(\pi)$.

Experiment from Swaminathan and Joachims

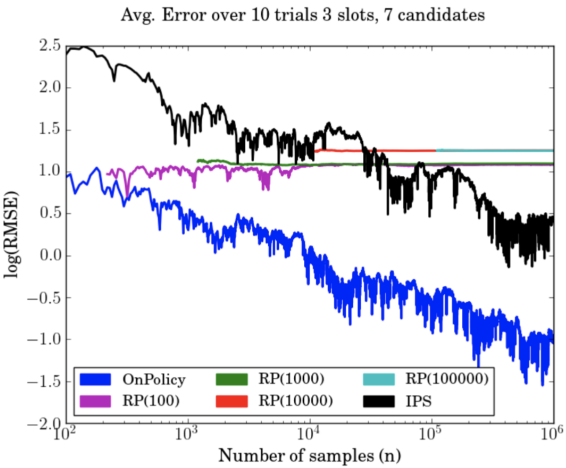


- IPS = inverse propensity scoring = importance weighted

From https://www.cs.cornell.edu/~adith/CfactSIGIR2016/Evaluation1.pdf page 38.

The IPS estimator (same as our IW estimator) has a lot more variance (see the large variation with fewer samples), but converges to the right thing.

Experiment from Swaminathan and Joachims

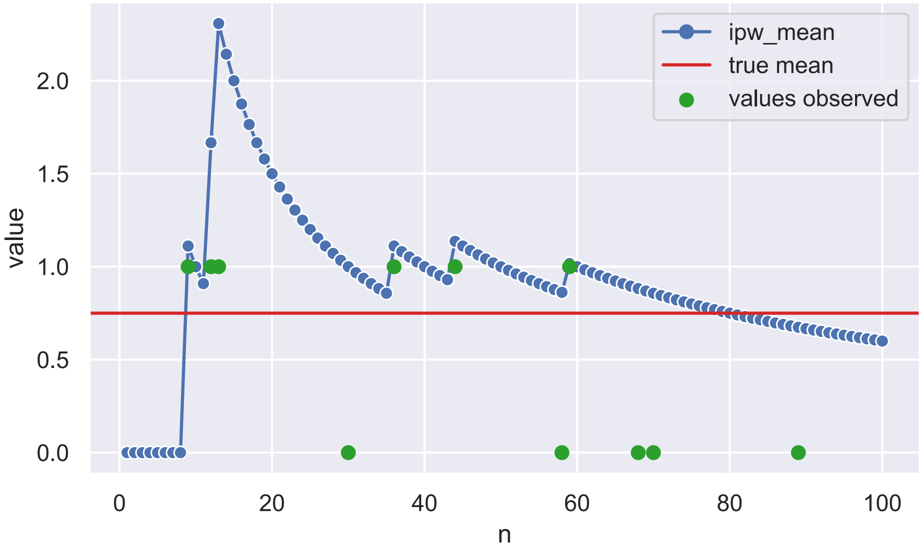


- IPS = inverse propensity scoring = importance weighted

From https://www.cs.cornell.edu/~adith/CfactSIGIR2016/Evaluation1.pdf page 38.

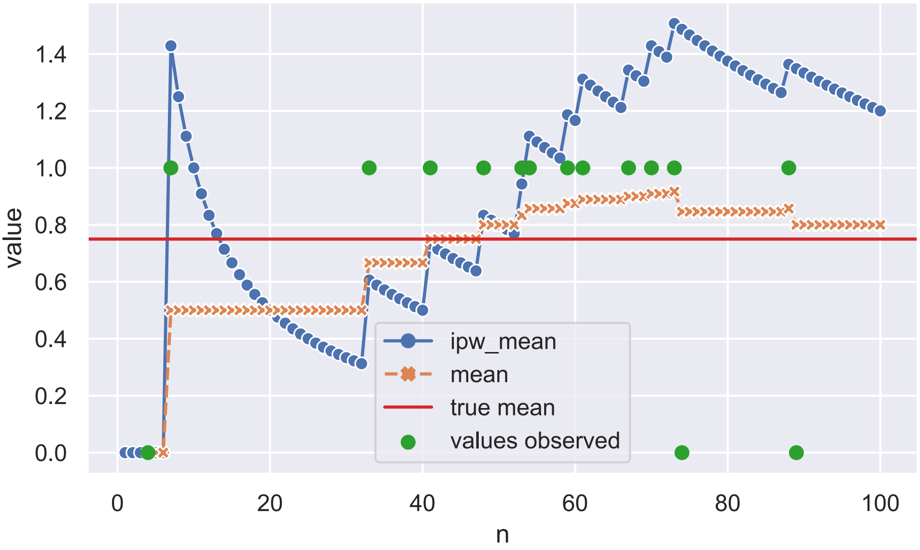
Self-normalized IW value estimators

IPW for MCAR example visualized



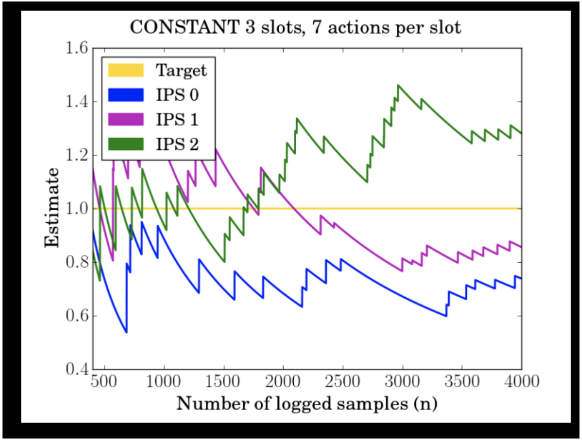
- The green dots represent observed values of Y_i.
- We can see that we had no observations of Y until $Y_9 = 1$.
- The horizontal red line shows the true mean of the Y_i's.
- The blue dots show $\hat{\mu}_{\text{ipw}}$ as n increases.
- Note the large jumps in $\hat{\mu}_{\text{ipw}}$ whenever we get an observation with $Y_i = 1$. This is because each observed Y_i is scaled up by an inverse propensity weight of 10.
- Also note that between observations with $Y_i = 1$, ipw_mean decays like $1/n$ towards 0.

IPW mean for "too many" observations



Simple experiment with $R_i \equiv 1.0$

- Swaminathan and Joachims try IW value estimator with $R_i \equiv 1.0$.



- 3 different sample paths

From https://www.cs.cornell.edu/~adith/CfactSIGIR2016/Evaluation2.pdf page 2.

- In these simulations, we try out the IW value estimator when all rewards are always 1.0.
- Ideally, our reward estimates would be very close to 1.0!
- Here are 3 runs of this, all show poor performance.
- The performance is exactly like that of the IPW mean in the missing data setting.

Self-normalized IW value estimator

Definition

The **self-normalized importance-weighted (IW) value estimator** for policy π, based on logged bandit feedback $(X_1, A_1, R_1(A_1)), \ldots, (X_n, A_n, R_n(A_n))$ with actions chosen under static logging policy π_0 is

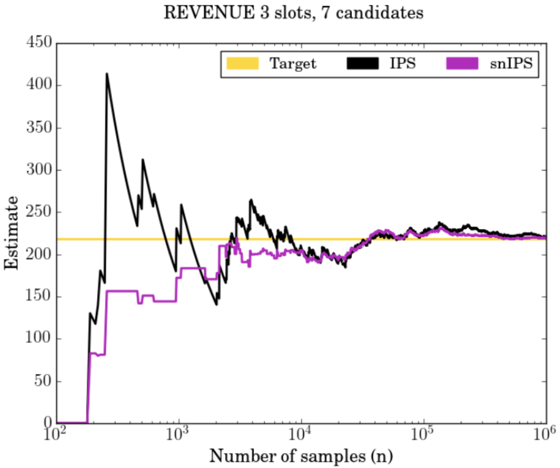
$$\hat{V}_{\text{sn\_iw}}(\pi) = \frac{\sum_{i=1}^{n} W_i R_i(A_i)}{\sum_{i=1}^{n} W_i},$$

where the **importance weights** are defined as

$$W_i := \frac{\pi(A_i \mid X_i)}{\pi_0(A_i \mid X_i)}.$$

- What happens to $\hat{V}_{\text{sn\_iw}}(\pi)$ when received rewards are always 1?

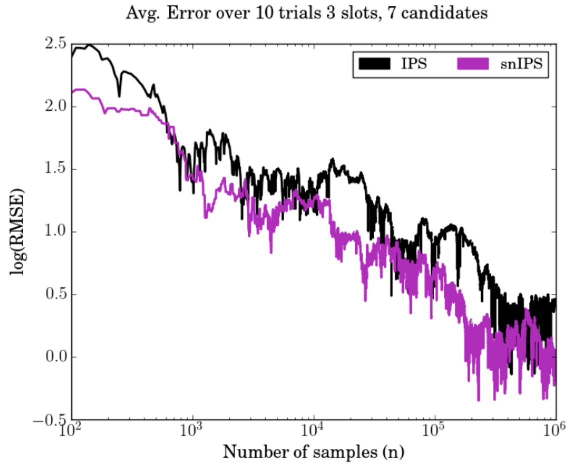
Self-normalized IW vs IW



From https://www.cs.cornell.edu/~adith/CfactSIGIR2016/Evaluation2.pdf page 6.

- Just as in the missing data setting, self-normalization can significantly reduce the variance.

Self-normalized IW vs IW



From https://www.cs.cornell.edu/~adith/CfactSIGIR2016/Evaluation2.pdf page 6.

References

Notation and Terminology

- A lot of our notation is based on [BWRB20], though we align with [SB18] and most probability textbooks in our use of capital letters to indicate random variables and lower case letters for the values they can take.
- Terminology-wise, what we call the importance weighted (IW) estimator is called the inverse propensity score (IPS) estimator in [JSdR18]. We call it the importance-weighted estimator to make the analogy to the importance-weighted empirical risk, which we defined in the module on covariate shift. Our terminology is also used in [BWRB20].

References I

[BWRB20] David Brandfonbrener, William F. Whitney, Rajesh Ranganath, and Joan Bruna, *Bandit overfitting in offline policy learning*, CoRR (2020).

[CL11] Olivier Chapelle and Lihong Li, *An empirical evaluation of thompson sampling*, Proceedings of the 24th International Conference on Neural Information Processing Systems (Red Hook, NY, USA), NIPS'11, Curran Associates Inc., 2011, pp. 2249–2257.

[DLL11] Miroslav Dudík, John Langford, and Lihong Li, *Doubly robust policy evaluation and learning*, Proceedings of the 28th International Conference on International Conference on Machine Learning (Madison, WI, USA), ICML'11, Omnipress, 2011, pp. 1097–1104.

References II

[JSdR18] Thorsten Joachims, Adith Swaminathan, and Maarten de Rijke, *Deep learning with logged bandit feedback*, 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, OpenReview.net, 2018.

[SB18] Richard S. Sutton and Andrew G. Barto, *Reinforcement learning: An introduction*, A Bradford Book, Cambridge, MA, USA, 2018.

[SJ15] Adith Swaminathan and Thorsten Joachims, *The self-normalized estimator for counterfactual learning*, Advances in Neural Information Processing Systems 28 (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), Curran Associates, Inc., 2015, pp. 3231–3239.

[WBBJ19] Lequn Wang, Yiwei Bai, A. Bhalla, and T. Joachims, *Batch learning from bandit feedback through bias corrected reward imputation*, ICML Workshop on Real-World Sequential Decision Making, 2019.