Modern Recommendation for Advanced Practitioners
Part II. Recommendation as Policy Learning

Authors: **Flavian Vasile, David Rohde, Amine Benhalloum, Martin Bompaire Olivier Jeunen & Dmytro Mykhaylov**

December 5, 2019

Course

- **Part I. Recommendation via Reward Modeling**
    - I.1. Classic vs. Modern: Recommendation as Autocomplete vs. Recommendation as Intervention Policy
    - I.2. Recommendation Reward Modeling via (Point Estimation) Maximum Likelihood Models
    - I.3. Shortcomings of Classical Value-based Recommendations
- **Part II. Recommendation as Policy Learning** Approaches
    - II.1. Policy Learning: Concepts and Notations
    - II.2. Fixing the issues of Value-based Models using Policy Learning
        - *Fixing Covariate Shift using Counterfactual Risk Minimization*
        - *Fixing Optimizer's Curse using Distributional Robust Optimization*
    - II.3. Recap and Conclusions

II.1. Policy Learning: Concepts and Notations

Acting using Policies vs. Acting using Value Models

Previously, we covered *Likelihood-based Models of Bandit Feedback*: we learnt how to create a reward model function of the pair *(user state, recommendation)*.

A *Value Model-based solution* computes for each user the predicted reward for all potential recommendation and then take the maximizing action (the *argmax*).

Acting using Policies vs. Acting using Value Models

We now look at a different way to choose actions, which is to directly learn a decision function that maps user states to actions, based on past rewards.

We call the mapping function a *policy*, and the direct approaches for optimizing it, *Policy Optimization* or *Policy Search* methods.

Value-based vs. Policy-based Methods. An Analogy.

- **Value-based models are like a driving instructor:** every action of the student is evaluated.
- **Policy-based models are like a driver:** it is the actual driving policy how to drive a car.

Figure 1: Instructor vs. Driver

Policy-based Methods: Contextual Bandits and Reinforcement Learning

Two main approaches:

- **Contextual Bandits** are the class of policy models where the state is not dependent on previous actions but only on current context, and the reward is immediate
- **Reinforcement Learning** solves the more complex policy learning problem where the state definition takes into account the previous actions and where the reward can be delayed, which creates additional credit assignment issues.

Contextual Bandits vs. Reinforcement Learning

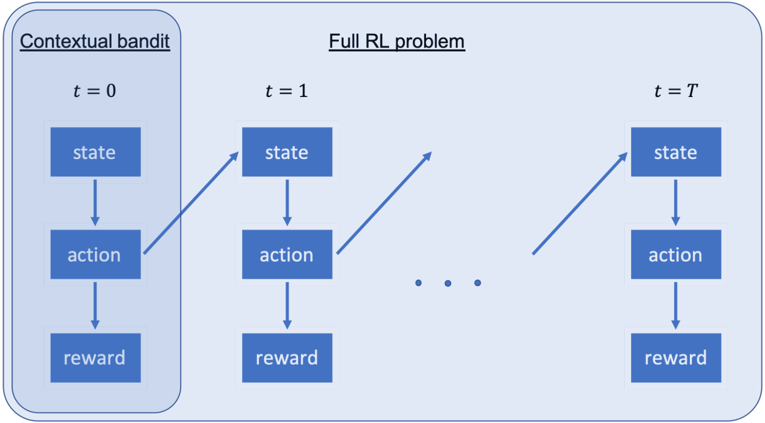


Figure 2: Contextual Bandits vs. RL

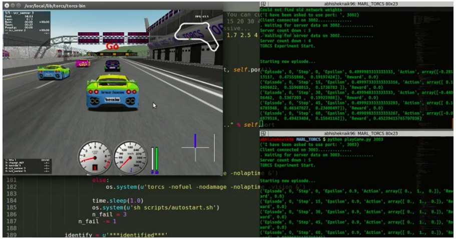
Contextual Bandits vs. Reinforcement Learning for Reco

Contextual Bandits (CB) are quite a good match for learning recommendation policies since, in most cases, each recommendation has an independent effect conditioned on the user state (context). In this case, we do not have to optimize for sequences of recommendations.

By making this simplifying assumption, we can *represent the user state strictly by looking at her historical organic activity*, that can be mapped to her naturally occurring interests.

Contextual Bandits vs. Reinforcement Learning for Reco

Reinforcement Learning (RL), though a very nice framework, might be at this moment too ambitious/too immature to be directly employed in live decision-making systems. So far, its applications are limited to fields that have good offline simulators, such as games and *possibly* self-driving cars.


/usr/local/lib/torcs/torcs-bin
GO
abhisheknaik96: MARL_TORCS 80x23
abhisheknaik96: MARL_TORCS 80x23

```
Could not find old network weights
('I have been asked to use port: ', 3002)
Client connected on 3002..............
. Waiting for server data on 3002..............
Server count down : 5
. Waiting for server data on 3002..............
Server count down : 4
TORCS Experiment Start.


Starting new episode...

('Episode', 0, 'Step', 0, 'Epsilon', 0.4999983333333333, 'Action', array([-0.281
13117,  0.47551866,  0.19197424]), 'Reward', 0.0)
('Episode', 0, 'Step', 15, 'Epsilon', 0.4999733333333316, 'Action', array([ 0.1
0406622,  0.55968815,  0.1738783 ]), 'Reward', 0.0)
('Episode', 0, 'Step', 30, 'Epsilon', 0.4999483333333333, 'Action', array([-0.440
06462,  0.5367203 ,  0.19925988]), 'Reward', 0.0)
('Episode', 0, 'Step', 45, 'Epsilon', 0.4999233333333283, 'Action', array([ 0.1
6785948,  0.46147027,  0.23600497]), 'Reward', 0.0)
('Episode', 0, 'Step', 60, 'Epsilon', 0.4998983333333267, 'Action', array([-0.0
6979516,  0.49423404,  0.15041162]), 'Reward', 0.45239415765707036)
```

```
abhisheknaik96 MARL_TORCS $ python playGame.py 3003
('I have been asked to use port: ', 3003)
Client connected on 3003..............
. Waiting for server data on 3003..............
Server count down : 5
TORCS Experiment Start.

Starting new episode...
('Episode', 0, 'Step', 0, 'Epsilon', 0.9, 'Action', array([ 0.,  1.,  0.]), 'Rew
ard', 0.0)
('Episode', 0, 'Step', 15, 'Epsilon', 0.9, 'Action', array([ 0.,  1.,  0.]), 'Re
ward', 0.0)
('Episode', 0, 'Step', 30, 'Epsilon', 0.9, 'Action', array([ 0.,  1.,  0.]), 'Re
ward', 0.0)
('Episode', 0, 'Step', 45, 'Epsilon', 0.9, 'Action', array([ 0.,  1.,  0.]), 'Re
ward', 0.0)
('Episode', 0, 'Step', 60, 'Epsilon', 0.9, 'Action', array([ 0.,  1.,  0.]), 'Re
```

```
                else:
                    os.system(u'torcs -nofuel -nodamage -nolaptime -vision &')

                time.sleep(1.0)
                os.system(u'sh scripts/autostart.sh')
                n_fail = 3
            n_fail -= 1

        identify = u'***identified***'
```

Figure 3: RL for simulated tasks (Santara et al., 2018)

Img Credits: MADRaS: A Multi-Agent DRiving Simulator (Santara et al., 2018)

Contextual Bandits vs. Reinforcement Learning for Reco

In order to deploy RL in real-world applications where exploration/learning from the environment is costly(slow), such as the case of Recommendation Systems, we need to either:

- **Build simulators** of the true environment (aka RecoGym)
- **Work on SafeRL** (RL with bounds on maximum disappointment)

For the rest of the course, we will concentrate on the Contextual Bandits case. To make things more concrete, we will start with some notations!

Policy Methods. Notations.

Context, Action, Policy

- $x \in$ arbitrary *contexts* drawn from an unknown distribution ν
- $y \in$ denote the *actions* available to the decision maker
- A *policy* is a mapping $\pi :\rightarrow ()$ from the space of contexts to probabilities in the action space. For a given (context, action) pair (x, y), the quantity $\pi(y|x)$ denotes the probability of the policy π to take the action y when presented with the context x.

Value, Risk and Risk Estimators:

- True action reward/utility of y in the context x: $\delta(x, y)$
- Cost/loss: $c(x, y) \triangleq -\delta(x, y)$
- Policy Risk:

$$R(\theta) \triangleq \mathbb{E}_{x \sim \nu, y \sim \pi_\theta(\cdot|x)} [c(x, y)] \tag{1}$$

  If we replace by P the joint distribution over (states,actions) (x,y) pairs, we have that: $R(\theta) \triangleq \mathbb{E}_{(x,y) \sim P} [l(x, y)]$

Empirical Risk Minimization

- Empirical Risk Estimator:

$$\hat{R}_n(\theta) \triangleq \frac{1}{n} \sum_{i=1}^{n} c_i \tag{2}$$

- **Empirical Risk Minimization (ERM)** / Sample Average Aproximation (SAA):

$$\hat{\theta}_n^{ERM} \triangleq \underset{\theta \in \Theta}{\operatorname{argmin}} \hat{R}_n(\theta) \tag{3}$$

Predicted vs. True Policy Risk

Because in real-world we always deal with finite samples, there exists always a gap between our current estimate of the risk of a policy and its true risk:

- **True Risk** of a policy: $R(\hat{\theta}_n) = E_{(x,y)\sim P}[c(x, y; \hat{\theta}_n)]$
- **Predicted Risk** of a policy: $\hat{R}_n(\hat{\theta}_n) = \frac{1}{n} \sum_{i=1}^{n} c_i$

Predicted vs. True Policy Risk

How should we think about these two quantities for the case of RecSys?

- **Predicted Risk:** The performance metrics collected from offline experiments using off-policy estimators
- **True Risk:** The online A/B test results using on-policy measurement (still coming from a finite sample, but of much bigger size and lower variance, being on-policy)

Evaluating Policies: Regret, Disappointment

In order to evaluate the performance of our policies, two quantities are critical:

- **Regret:** The difference in true risk between the policy $\hat{\theta}_n$ being evaluated and the true best policy $\theta^\star$ (of course, we want to minimize Regret):

$$Reg(\hat{\theta}_n) = R(\hat{\theta}_n) - R(\theta^\star) \tag{4}$$

- **Disappointment/Post-decision surprise:** The difference between the true risk R and the predicted risk $\hat{R}_n$ of the policy being evaluated. As we pointed out in the previous slides, this is the difference between *offline results (what you promised your manager)* and *online results (what actually happens when you actually try it)*:

$$Dis(\hat{\theta}_n) = R(\hat{\theta}_n) - \hat{R}_n(\hat{\theta}_n) \tag{5}$$

Evaluating Policies: Regret, Disappointment

A lot of the traditional theoretical work in Bandits and RL concentrated on providing bounds and guarantees on **Regret**, but more recently researchers have started to look more carefully at bounding **Disappointment**.

In our course, we will show why this is important and how it can improve our overall performance.

Switching from Value-based Reco to Policy-based Reco

Switching from Value-based Reco to Policy-based Reco

From the p.v of the final decision-making system, we are replacing:

The classical approach of Reco as a deterministic recommendation policy that:

- Computes using ERM the contextual pClick of all actions
- Returns the action that satisfies argmax(pClick)

with:

The more modern approach of Reco as a stochastic policy that:

- Directly puts more probability on contextual actions that worked well in the past.

Note: Because Value-based methods are essentially a wrapper around ERM-based models for decision making, we **will refer to them as ERM decision methods** and to their associated best estimated policies by: $\hat{\theta}_n^{ERM}$.

Okay, but why should we switch?

Q: Okay, so policy-based models are more modern and RL is fashionable, but why should we switch from likelihood-based models?

A: Well, as we saw in Part I, it turns out that value-based approaches can suffer from a number of shortcomings that policy-based methods can address.

We will take a look at ways of fixing these shortcomings in the this part of the course.

II.2 Fixing the issues of Value-based Models using Policy Learning

II.2.1. Fixing Covariate Shift using Counterfactual Risk Minimization

Moving away from Direct Reward Modeling

In Section 2, we presented the issue of the Covariate Shift, where the fact that the *training* and the *test* distributions are different can affect the final performance of our ERM-based solutions.

In order to avoid this, one solution would be to map the training data into the test distribution using *the IPS trick*.

The question then becomes:

What is the target distribution we want to map the bandit data into?

Fixing Covariate Shift using IPS for Value-based Methods

In Value-based models, we evaluate all actions given the state in order to compute argmax, so we need to be uniformly good everywhere. Therefore, we should **map the π_0 data into π_{unif} as the new training distribution.**

- Instead of looking at fitting a model for: $y_{ij} = r_{ij} \times \pi_0(j|i)$ we will directly fit a model for $y_{ij} = r_{ij} \times \frac{\pi_{unif}}{\pi_0}$, to simulate a distribution where all arms are equally exposed.

Fixing Covariate Shift using IPS for Policy Learning Methods

In the case of policy learning, the application of the IPS trick is straight-forward since we can compute for any target policy its estimated risk based on a different logging policy by using the **Counterfactual Risk Estimator**:

$$R^{CRM}(\theta) = \mathbb{E}_{x \sim \nu, y \sim \pi_\theta(x)} \left[c(x, y) \right] = \mathbb{E}_{x \sim \nu, y \sim \pi_0(x)} \left[c(x, y) \frac{\pi_\theta(y|x)}{\pi_0(y|x)} \right], \tag{6}$$

Fixing Covariate Shift using IPS for Policy Learning Methods

- Empirical Counterfactual Risk Estimator:

$$\hat{R}_n^{CRM}(\theta) \triangleq \frac{1}{n}\sum_{i=1}^{n} c_i \frac{\pi_\theta(y|x)}{\pi_0(y|x)} \tag{7}$$

Note: $\hat{R}_n^{CRM}$ is an unbiased estimator of the true risk of the policy $\pi_\theta(y|x)$, however it has unbounded variance due to the propensity ratio between the acting and the logging policy.

Counterfactual Risk Minimization (CRM)

By minimizing the re-weighted risk estimator, we obtain the following **Counterfactual Risk Minimization (CRM)** objective:

$$\hat{\theta}_{\boldsymbol{n}}^{CRM} \triangleq \underset{\theta \in \Theta}{\operatorname{argmin}} \hat{R}_{\boldsymbol{n}}^{CRM}(\theta) \tag{8}$$

Definition: We denote the set of contextual bandit methods that optimize for the CRM objective as **Off-Policy Contextual Bandits**.

Note: Off-Policy $=$ Counterfactual

Clipping IPS weights

- **Issue:** The new policy can explore "almost" unknown region of the previous policy i.e. $\pi_0(y|x)$ can be very small compared to $\pi_t(y|x)$, this lead to very high variance of the estimator.
- **Solution:** Clip the weights by replacing $\frac{\pi_t(y|x)}{\pi_0(y|x)}$ by $\min(M, \frac{\pi_t(y|x)}{\pi_0(y|x)})$, where M is a well chosen constant, we are trading some variance with some bias.

Self Normalized IPS

- **Issue 1:** The estimator built previously is sensitive to the cost scaling i.e. if we add a constant C to the cost, the *ERM* estimator will also be translated by C but not the IPS weighted estimator.
- **Issue 2:** This scale sensitivity can lead to propensity overfitting e.g. Imagine we are trying to minimize eq. 6 and that the costs c_i are positive. A solution that completely avoids the training data i.e. $\pi_\theta(y|x) = 0$ achieves the global loss minimum !
- **Solution:** Use the Self-Normalized IPS estimator (we are again trading variance with bias by using a control variate)

$$R^{SNIPS}(\theta) = \frac{\sum_{i=1}^{N} c_i \frac{\pi_\theta(a_i|\mathbf{x_i})}{\pi_0(a_i|\mathbf{x_i})}}{\sum_{i=1}^{N} \frac{\pi_\theta(a_i|\mathbf{x_i})}{\pi_0(a_i|\mathbf{x_i})}}$$

Counterfactual Risk Minimization (CRM) for Recommendation

In the case of Recommendation, we do not want to allow the agent to directly act according to the target policy without supervision, so we do not allow continuous policy improvements online.

In general, we learn the new policy based on batch logged data and release it using an A/B test against the current policy.

Exercice 4: Contextual Bandit

A Simple Off-Policy Contextual Bandit Formulation

We want to learn the policy to choose the best action among all a_j given a state s_i.

We can parametrize this policy with **softmax**:

$$\pi_\theta(a|s_i) = \left(\frac{e^{<a_0, s_i>}}{\sum_{j'} e^{<a_{j'}, s_i>}} \;,\; \dots \;,\; \frac{e^{<a_j, s_i>}}{\sum_{j'} e^{<a_{j'}, s_i>}} \;,\; \dots \right)$$

And the best action is sampled from this policy with a multinomial distribution

$$a_i = \text{Multinomial}\big(\pi_\theta(a|s_i)\big)$$

Vanilla Contextual Bandit: Formulation

We want to maximize the number of clicks we would have got with π_θ. Namely we want to optimize

CRM Objective:

$$\hat{\theta}_n^{CRM} = \arg\max_{\theta} \sum_{i=1}^{n} \frac{\pi_\theta(a_i|s_i)}{\pi_0(a_i|s_i)} \cdot \mathrm{click}_i$$

Vanilla Contextual Bandit with Clipped weights

To control the variance we can the probability ratio to a value M

Clipped CRM Objective:

$$\hat{\theta}_n^{CRM} = \arg\max_{\theta} \sum_{i=1}^{n} \min(\frac{\pi_\theta(a_i|s_i)}{\pi_0(a_i|s_i)}, M) \cdot \text{click}_i$$

Log Contextual Bandit

What if we try to cast the previous objective to be closer to a weighted log-likelihood ?

We can lower bound its log using Jensen's inequality:

$$\log\left(\sum_{i=1}^{n}\frac{\pi_\theta(a_i|s_i)}{\pi_0(a_i|s_i)}\mathrm{click}_i\right) \geq \frac{1}{\sum_{i=1}^{n}\frac{\mathrm{click}_i}{\pi_0(a_i|s_i)}}\sum_{i=1}^{n}\frac{\mathrm{click}_i}{\pi_0(a_i|s_i)}\log\left(\pi_\theta(a_i|s_i)\right) + \log\left(\sum_{i=1}^{n}\frac{\mathrm{click}_i}{\pi_0(a_i|s_i)}\right)$$

Log Contextual Bandit

$\hat{\theta}$ that maximizes this lower bound also maximizes

$$\sum_{i=1}^{n} \frac{\text{click}_i}{\pi_0(a_i|s_i)} \log\left(\pi_\theta(a_i|s_i)\right).$$

Since we have parameterized the policy with softmax,

$$\pi_\theta(a_j|s_i) = \frac{e^{<a_j, s_i>}}{\sum_{j'} e^{<a_{j'}, s_i>}},$$

this objective is the log-likelihood of a multinomial logistic regression where each observation has been weighted by $\frac{\text{click}_i}{\pi_0(a_i|s_i)}$.

Note that this reverts to optimize log-likelihood under a uniform sampling (instead of π_0 sampling with CRM or π_θ with true ERM).

Log Contextual Bandit

Let's rephrase what we want to optimize:

A **multinomial** logistic regression where:

- **labels** are the actions that have led to a click.
- **features** are the user states when the action was taken.
- each observation has been **weighted** by $\frac{\text{click}_i}{\pi_0(a_i|s_i)}$, hence, we end up considering only clicked observations.

Very easy to derive from an existing logistic regression implementation.

Contextual Bandit and LogCB. Exercises.

You can open this notebook **(click here!)**, where a “ContextualBandit“ agent has been implemented to illustrate the two previous formulations.

- Optimizing directly the IPS weighted objective
- Optimizing the lower bound of its log, that reduces to a logistic regression with weighted examples.

Contextual Bandit and LogCB. Conclusions.

We saw that the Contextual Bandits can be a valid contender of the likelihood-based agents, reaching a slightly better CTR. Furthermore, we saw that a simple bandit solution is not much harder to implement or to understand than a typical supervised model. So hopefully, this demystified a bit the topic.

Q: What's next? Can we do even better?

A: Yes, and it comes back to fixing another of the issues of value-based models.

3.2. Fixing Optimizer's Curse using Distributional Robust Optimization

CRM and the Optimizer’s Curse

CRM and the Optimizer's Curse

Remember that we criticized the value-based models by pointing out two things:

- **The Covariate Shift** - We solved it by moving to an IPS-based objective
- **The Optimizer's Curse** - *How do we fix it?*

CRM and the Optimizer's Curse. The Bad News.

It's interesting to note that, by using IPS for our offline reward estimators to counter the Covariate Shift problems, and therefore switching to a CRM objective, **the variance of the arms/actions pulled rarely by the existing policy will get amplified by the IPS term**, so it is even more likely that the reward of a rare action will be hugely over-estimated.

By fixing Covariate Shift, we accentuate the Optimizer's curse!

CRM and the Optimizer's Curse

Using the Disappointment-based definition of Optimizer's curse, we want to be able to have a CRM-based method that is able to **bound the probability of incurring disastrous outcomes**, aka:

$$\lim_{n \to \infty} P\left(Dis(\hat{\theta}_n) \geq 0\right) \leq \delta, \tag{9}$$

Fixing Optimizer's Curse

The source of the Optimizer's curse is that at decision time, **we do not take into account the uncertainty level of the reward/risk estimator, but only its expectation** - which is what ERM optimizes for.

Can we do something about it?

Fixing Optimizer's Curse

There are multiple ways to incorporate in our decision making:

- **Go fully Bayesian** on it
- Use again modeling approaches from Portofolio Optimization, that **directly address the problem of bounding disappointment**.

Bounding Disappointment with Robust Risk

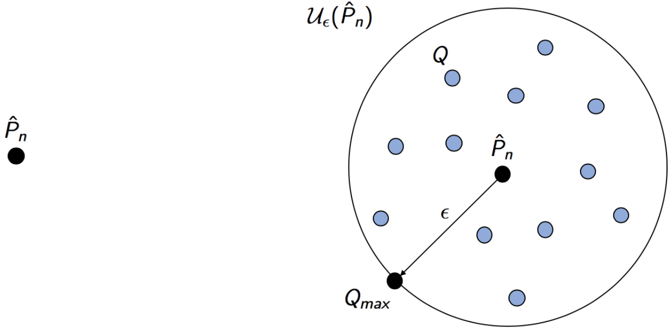
One way to fix the ERM limitations is to **treat the empirical training distribution over (state,action) pairs with skepticism**.

The robust approach: We are going to replace the empirical sample with an uncertainty set $_{\epsilon}()$ of distributions around that are *consistent* with the data (where $\epsilon > 0$ is a parameter controlling the size of the uncertainty set $\mathcal{U}_{\epsilon}()$).

This gives rise to the *Robust Risk* (bold letters indicate robust risk and estimators):

$$\theta \triangleq \max_{Q \in_{\epsilon}()} \mathbb{E}_{(x,y)\sim Q}[\ell(x,y;\theta)]. \tag{10}$$

Robust Risk



$$\hat{R}_n(\theta) \triangleq \mathbb{E}_{(x,y)\sim\hat{P}_n}\,[l(x,y;\theta)]$$

$$\hat{\mathbf{R}}_n^{\mathcal{U}}(\theta) \triangleq \max_{Q\in\mathcal{U}_\epsilon(\hat{P}_n)} \mathbb{E}_{(x,y)\sim Q}[\ell(x,y;\theta)].$$

Figure 4: Empirical Risk vs. Empirical Robust Risk

Robust Risk. Choice of divergences

F-divergences: In probability theory, an **f-divergence** is a function $D_f(P||Q)$ that measures the difference between two probability distributions P and Q.

Intuition: Think of the F-divergence as an average, weighted by the function f, of the odds ratio given by P and Q. For the ones familiar with KL-divergence, this is a generalization of the concept.

Examples: KL-divergence, Reverse KL-divergence, Chi-squared distance

Distributional Robust Optimization (DRO)

Minimizing this quantity w.r.t to θ yields the *general DRO program*:

$$\hat{\theta}_n^{\text{rob}} \triangleq \text{argmin}_{\theta \in \Theta} \theta = \underset{\theta \in \Theta}{\text{argmin}} \max_{Q \in \mathcal{U}_\epsilon(P_n)} \mathbb{E}_{(x,y)\sim Q}[\ell(x, y; \theta)]. \tag{11}$$

Fixing the asymptotic version of Optimizer's Curse with DRO

DRO's Risk Adversiveness provides a bound on Disappointment: Given the DRO objective, and uncertainty sets $\mathcal{U}_\epsilon()$ defined using *coherent f-divergences*, one can prove asymptotic performance guarantees - see Duchi et al. (2016):

$$\lim_{n\to\infty} P\left(Dis(\hat{\theta}_n^{DRO}) \geq 0\right) \leq \delta, \tag{12}$$

by comparison with **ERM's Asymptotic Risk Neutrality:**

$$\lim_{n\to\infty} P\left(Dis(\hat{\theta}_n^{ERM}) \geq 0\right) = 1/2, \tag{13}$$

The DRO solution is asymptotically consistent

In the limit, the DRO policy is minimizing both **Disappointment** and **Regret**:

$$\lim_{n \to \infty} Dis(\hat{\theta}_n^{DRO}) = 0 \tag{14}$$

$$\lim_{n \to \infty} Reg(\hat{\theta}_n^{DRO}) = 0 \tag{15}$$

Robust solutions are consistent under (essentially) the same conditions required for that of sample average approximation Duchi et al. (2016).

DRO vs. ERM

Unlike ERM, DRO bounds the probability of disappointment while preserving the asymptotic ERM behaviour of minimizing both **Regret** and **Disappointment**. That means that the **DRO objective can recover the optimal policy** as the sample size n goes to infinity, but it does it while **going through a safer learning route than ERM**.

Disclaimer: This behaviour is in the limit, so the DRO objective might lead to a safer but longer convergence path than ERM to the optimal policy (see ongoing work on determining finite sample guarantees on regret and disappointment).

DRO for the CRM Problem

DRO-CRM: DRO for Counterfactual Risk Minimization

- We introduced the CRM objective as a way to alleviate the *Covariate Shift* problems in Recommendation and we have observed that the CRM objective can accentuate the *Optimizer's Curse* by amplifying the variance of rare actions
- We also have shown that DRO can address the *Optimizer's Curse* by penalizing variance.
- It is therefore quite natural to apply DRO in the CRM context:

The DRO-CRM objective:

$$\hat{\theta}_n^{\text{CRM-rob}} \triangleq \operatorname*{argmin}_{\theta \in \Theta} \max_{Q \in \mathcal{U}_\epsilon(P_n)} \mathbb{E}_{(x,y)\sim Q}[\ell^{CRM}_{(x,y;\theta)}]. \tag{16}$$

DRO-CRM objective. Impact on Policy Learning

In the context of the CRM policy optimization, the DRO objective will **penalize policies that are far** from the logging policy.

The minimization of the robust risk based on **any coherent F-divergences** is *asymptotically equivalent* with penalizing the empirical risk by the empirical risk variance - see Duchi et al. (2016)

The minimization of the robust risk based on **the Chi-squared** is *exactly equivalent* with penalizing the empirical risk by the empirical risk variance - see Duchi et al. (2016)

DRO-CRM with coherent F-divergences. Algorithms

POEM: The special case of using of Chi-squared distance for CRM optimization was introduced in Swaminathan and Joachims (2015). The authors proposed an optimization method under the name of **Policy Optimizer for Exponential Models (POEM)**.

DRO-CRM-KL: The full generalization to coherent F-divergences was proposed in Faury et al. (2019) along with an algorithm for exponential policy optimization with KL divergence.

POEM

The POEM objective:

$$^{\varphi}\theta = \hat{R}_n(\theta) + \sqrt{\epsilon V_n(\theta)} + \alpha_n(\theta), \tag{17}$$

where:

- $\hat{R}_n(\theta)$ is the counterfactual risk
- $V_n(\theta)$ denotes the empirical variance of the quantities $c_i \frac{\pi_\theta(y_i|x_i)}{p_i}$.
- $\alpha_n(\theta) \longrightarrow 0$

Exercise: POEM - Policy Optimizer for Exponential Models

POEM - Policy Optimizer for Exponential Models

In our previous example we showed an IPS weighted objective for logistic regression. Now we are going to add “Sample Variance Penalization“ to counter the instability of this objective and optimizer’s curse.

POEM - Policy Optimizer for Exponential Models

Let’s open CRM notebook: **click here!**

Beyond DRO-CRM in the case of Recommendation

DRO-CRM is a very general policy learning approach, but in the case of Recommendation, we have one more source of information, which is **the organic user behaviour**.

The advantage of organic behaviour is that it does not suffer from the bandit problems, since *we are only training on it, but not acting on it* (and therefore we are not triggering the Optimizer's Curse and the Covariate Shift issues).

Section 4. Recap and Conclusions

To recap, the future is counterfactually robust!

- We saw that by switching from an ERM to a DRO objective we can *fix Optimizer's Curse*.
- We saw that by switching from an ERM to a CRM objective we can *fix Covariate Shift*.

In the end we merged the two changes to a obtain a new objective, namely **DRO-CRM**.

Organic and Bandit tasks. A Recap of Methods

Feedback	Framework	Task	Question
Organic	Sequence prediction	Predict $Proba(P_+ \mid U)$	What is the proba of P being observed next $(+)$, given U?
Organic	Classification	Predict $Proba(+ \mid U, P)$	What is the proba of the pair of U and P being observed?
Organic	Ranking	Rank $(U, P_+) \geq (U, P_{unk})$	Rank higher (U, P) observed pairs than unobserved ones
Bandit	Likelihood	Predict $Proba(Click \mid U, A)$	Predict probaClick for all (U,A) pairs
Bandit	Policy Search	Predict $\log Proba(A \mid U) \times \frac{Click}{\pi_0}$	Predict actions given user weighted by their obs. rw-clicks

Organic and Bandit tasks. A Recap of Methods

Feedback	Framework	Task	Predictor/Loss
Organic	Sequence prediction	Predict $Proba(P_+\|U)$	Softmax, CrossEntropy
Organic	Classification	Predict $Proba(+\|U, P)$	Sigmoid, BinCross
Organic	Ranking	Rank $(U, P_+) \geq (U, P_{unk})$	Sigmoid, RankingLoss
Bandit	Likelihood	Predict $Proba(Click\|U, A)$	Sigmoid, BinCross
Bandit	Policy Search	Predict $\log Proba(A\|U) \times \frac{Click}{\pi_0}$	Softmax, WCrossEntropy

Organic and Bandit tasks. A Recap of Methods

Feedback	Framework	Task	Algo
Organic	Sequence prediction	Predict $Proba(P_+\|U)$	W2V, RNN, TCN, ATTN
Organic	Classification	Predict $Proba(+\|U, P)$	LogReg
Organic	Ranking	Rank $(U, P_+) \geq (U, P_{unk})$	BPR
Bandit	Likelihood	Predict $Proba(Click\|U, A)$	LogReg
Bandit	Policy Search	Predict $\log Proba(A\|U) \times \frac{Click}{\pi_0}$	W-Multinomial LogReg

What we did not cover

- Exploration
- Bayesian approaches to reason about uncertainty

Thank You!

Questions?

References

Duchi, J., Glynn, P., and Namkoong, H. (2016). Statistics of robust optimization: A generalized empirical likelihood approach.

Faury, L., Tanielian, U., Vasile, F., Smirnova, E., and Dohmatob, E. (2019). Distributionally robust counterfactual risk minimization. *arXiv preprint arXiv:1906.06211*.

Santara, A., Naik, A., Ravindran, B., and Kaul, B. (2018). Madras: A multi-agent driving simulator.

References ii

Swaminathan, A. and Joachims, T. (2015). Counterfactual Risk Minimization: Learning from Logged Bandit Feedback. In *International Conference on Machine Learning*, pages 814–823.