NEURAL NETWORKS AND CAUSAL RECOMMENDATION

Part II. Policy Learning for Recommendation

Flavian Vasile, Martin Bompaire

Criteo AI Lab

June 26, 2019

Collaborators:

Special thanks to my colleagues:

- David Rohde
- Dmytro Mykhaylov
- Stephen Bonner

criteo

AI Lab

http://cail.criteo.com

Types of approaches to Recommendation

- **Classical Approach: Recommendation as auto-complete** methods:
  - Typically leverage *organic* user behaviour information (e.g. item views, page visits)
  - Frame the problem either as *missing link prediction/MF* problem, either as a *next event prediction, sequence prediction* problem
- **Upcoming approaches: Recommendation as intervention policy** methods:
  - Typically leverage *bandit* user behaviour information (e.g. ad clicks)
  - Frame the problem either as *click likelihood* problem, either as a *contextual bandit/policy learning* problem

Two courses on Recommendation at DS3 2019

- **Course I. CLASSICAL ALGORITHMS AND MATRIX FACTORIZATION - Recommendation as auto-complete** approaches
  - I.1. Recommendation via missing link prediction
  - I.2. Recommendation via next event prediction
- **Course II. NEURAL NETWORKS AND CAUSAL RECOMMENDATION - Recommendation as intervention policy** approaches
  - II.1. Recommendation via maximizing click likelihood
  - II.2. Recommendation as policy learning

Two courses on Recommendation at DS3 2019

- **Course I. CLASSICAL ALGORITHMS AND MATRIX FACTORIZATION - Recommendation as auto-complete** approaches
  - I.1. Recommendation via missing link prediction
  - I.2. Recommendation via next event prediction
- **Course II. NEURAL NETWORKS AND CAUSAL RECOMMENDATION - Recommendation as intervention policy** approaches
  - II.1. Recommendation via maximizing click likelihood
  - **We are here: II.2. Recommendation as policy learning**

The structure of our session:

- **Section 1:** Policy Learning: Concepts and Notations
- **Section 2:** Shortcomings of Empirical Risk Minimization (ERM): *Covariate Shift* and *Optimizer's Curse*
- **Section 3:** Fixing ERM using Policy Learning
  - *Fixing Covariate Shift: From ERM to Counterfactual Risk Minimization*
  - *Fixing Optimizer's Curse: From ERM to Distributional Robust Optimization*
  - Reco-specific Policy Learning methods: Using Organic Feedback
- **Section 4:** Recap and Conclusions

Getting started with RecoGym and Google Colaboratory

- Open your favourite browser, and go to the RecoGym repository:
  `github.com/criteo-research/reco-gym`
- Go to the DS3 branch, and open the slides on your laptop
  (they can be found in the 'slides' folder)
- URLs linking to Google Colaboratory notebooks for the hands-on sessions:
  - `https://colab.research.google.com/github/criteo-research/reco-gym/blob/DS3/Module%20%20IV.ipynb`
- Alternatively, clone the repository and run the notebooks locally.
- . . . you're all set!

Section 1. New 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 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

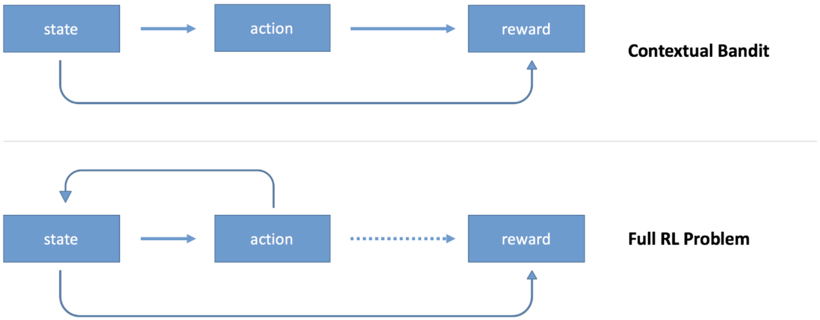


Figure 2: Contextual Bandits vs. RL

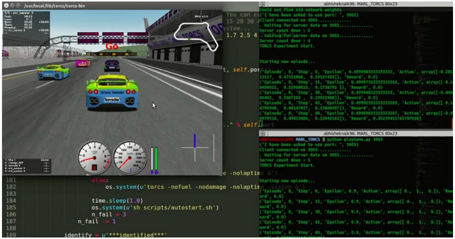
Contextual Bandits vs. Reinforcement Learning for Reco

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
TORCS
abhisheknaik96: MARL_TORCS 80x23
abhisheknaik96: MARL_TORCS 80x23

```
            else:
                os.system(u'torcs -nofuel -nodamage -nolaptim

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

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

```
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', 0.49997333333333316, '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', 0.49992333333333283, 'Action', array([ 0.1
6785948,  0.46147027,  0.23600497]), 'Reward', 0.0)
('Episode', 0, 'Step', 60, 'Epsilon', 0.49989833333333267, '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
```

Figure 3: RL for simulated tasks

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

- $x \in \mathcal{X}$ arbitrary *contexts* drawn from an unknown distribution ν
- $y \in \mathcal{Y}$ denote the *actions* available to the decision maker
- A *policy* is a mapping $\pi : \mathcal{X} \rightarrow \mathcal{P}\left(\mathcal{Y}\right)$ 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:

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

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

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):

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

Counterfactual/Off-Policy Risk and Estimators:

- Counterfactual Risk: In the case we want to estimate the risk of the policy $\pi_\theta(y|x)$ using data collected from a different policy - $\pi_0(y|x)$, we can use the *Inverse Propensity Scoring (IPS)* method to do it:

$$R^{CRM}(\theta) = \mathbb{E}_{x\sim\nu, y\sim\pi_\theta(x)}\left[c(x,y)\right] = \mathbb{E}_{x\sim\nu, y\sim\pi_0(x)}\left[c(x,y)\frac{\pi_\theta(y|x)}{\pi_0(y|x)}\right], \quad (4)$$

- Empirical Counterfactual Risk Estimator:

$$\hat{R}_n^{CRM}(\theta) \triangleq \frac{1}{n}\sum_{i=1}^{n} c_i \frac{\pi_\theta(y|x)}{\pi_0(y|x)} \quad (5)$$

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.

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):

$$Reg(\hat{\theta}_n) = R(\hat{\theta}_n) - R(\theta^\star) \tag{6}$$

- **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)*:

$$Dis(\hat{\theta}_n) = R(\hat{\theta}_n) - \hat{R}_n(\hat{\theta}_n) \tag{7}$$

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, 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 these shortcomings in the next part of the course.

Section 2. Shortcomings of Value-based Recommendations

2.1. The Covariate Shift Problem

Issue #1: The Covariate Shift Problem

The problem: We train our value model on one distribution of (user,action) / (x,y) pairs, but we test on another!

- Train on: $c(x, y) \times \pi_0(y|x)$
- Test on: $c(x, y) \times \pi_{new}(y|x)$

Since by definition we are learning a model of the rewards in order to do better recommendations, π_0 and π_{new} will always be different!

Covariate Shift. An Example

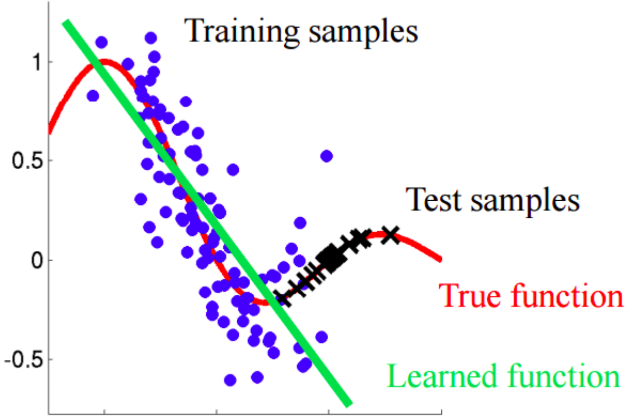


Figure 4: Train vs. Test: Blue vs. Black

Covariate Shift and Model Misspecification

Covariate Shift and Model Underfitting

- If we have a perfect predictor for $P(y|x)$, why should we care if $P(x)$ changes?
- Of course, $P(y|x)$ is not perfect and it is usually designed to underfit (to insure generalization through capacity constraints - limited degrees of freedom)
- In this case, we should allocate the model capacity on the portions of the $P(X)$ where the model will be tested

Covariate Shift and Recommendation

- **Note:** In the case of using value-based models for decision making (and in our case, Recommendation) we will evaluate all actions for all contexts so we will need to be good everywhere!

In Section 3, we will see how we are going to solve this, but first let's talk about the second issue!

2.2. The Optimizer's Curse

Issue #2: The Optimizer's Curse. A Story

Day1: Offline Evaluation Time (Predicted Reward)

- Imagine a decision problem, handed to us by our CEO, where we need to choose between **3 actions** and where, for each one of them, we have samples of historical rewards.
- In order to choose, we build empirical reward estimators for each of the 3 actions, we find the one that has the best historical performance and we go back to our CEO with the recommendation! After work, we congratulate ourselves and go to sleep happy!

Issue #2: The Optimizer's Curse. A Story

Day2: Online Evaluation Time (True Reward)

- However, the next day we meet with the CEO only to find out this was just a test of our decision-making capabilities, and that all of the sampled rewards for the 3 actions were **all drawn independently from the same distribution**.
- Due to the finite nature of our samples, each one of our 3 reward estimators made errors, some of them more negative (pessimistic), and some of them more positive (optimistic).
- Because when we decided, we selected the action with the highest estimate, we obviously favored the most optimistic of the 3, making us believe we optimized our decision, when we were just overfitting the historical returns. This effect is known at the **Optimizer's Curse**.

Issue #2: The Optimizer's Curse. A Story

Figure 1 The Distribution of the Maximum of Three Standard Normal Value Estimates

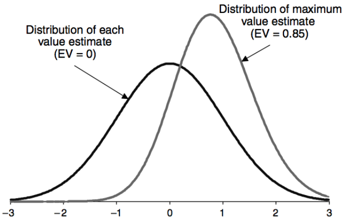
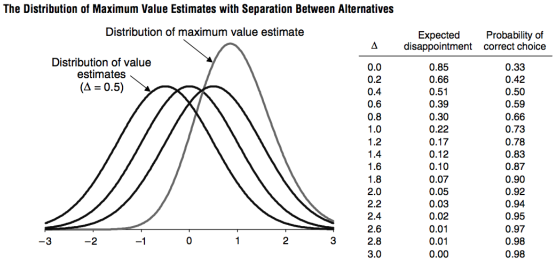


Figure 5: The distribution of rewards of argmax of 3 identical actions

Img Credits: The Optimizers Curse: Skepticism and Postdecision Surprise in Decision Analysis

Issue #2: The Optimizer's Curse. A Story



The Distribution of Maximum Value Estimates with Separation Between Alternatives

Δ	Expected disappointment	Probability of correct choice
0.0	0.85	0.33
0.2	0.66	0.42
0.4	0.51	0.50
0.6	0.39	0.59
0.8	0.30	0.66
1.0	0.22	0.73
1.2	0.17	0.78
1.4	0.12	0.83
1.6	0.10	0.87
1.8	0.07	0.90
2.0	0.05	0.92
2.2	0.03	0.94
2.4	0.02	0.95
2.6	0.01	0.97
2.8	0.01	0.98
3.0	0.00	0.98

Figure 6: The distribution of rewards of argmax of 3 actions with delta separation

Img Credits: The Optimizers Curse: Skepticism and Postdecision Surprise in Decision Analysis

The Optimizer's Curse: Definition

Going back to our original notations, we see that the difference between the *predicted risk* from **day1** and the *true risk* discovered in **day2**, is nothing else than the definition of **Disappointment/Post-decision surprise**.

Using this notation, we can describe the **Optimizer's Curse** as the fact that disappointment of ERM-based decisions is always positive:

$$Dis(\hat{\theta}_n^{ERM}) = R(\hat{\theta}_n^{ERM}) - \hat{R}_n(\hat{\theta}_n^{ERM}) \geq 0 \tag{8}$$

and, for the case when $\hat{\theta}_n^{ERM} \neq \theta^\star$:

$$Dis(\hat{\theta}_n^{ERM}) > 0 \tag{9}$$

Optimizer's Curse: Proof

PROOF: If we assume that $\hat{\theta}_n$ is the policy that minimizes the *unbiased* empirical risk estimator $\hat{R}_n$, then we have that:

$$Dis(\hat{\theta}_n) = R(\hat{\theta}_n) - \hat{R}_n(\hat{\theta}_n) \geq R(\hat{\theta}_n) - \hat{R}_n(\theta^\star) \geq R(\theta^\star) - \hat{R}_n(\theta^\star) = 0$$

Optimizer's Curse in the limit

The good news - ERM's Asymptotic Optimality: In the limit, the ERM policy is minimizing both **Disappointment** and **Regret**:

$$\lim_{n \to \infty} Dis(\hat{\theta}_n^{ERM}) = 0 \tag{10}$$

$$\lim_{n \to \infty} Reg(\hat{\theta}_n^{ERM}) = 0 \tag{11}$$

Optimizer's Curse in the limit

The bad news - ERM's Asymptotic Risk Neutrality a.k.a. The Optimizer's Curse: The empirical risk estimate $\hat{R}_n(\hat{\theta}_n)$ cannot be used as a *performance certificate* for the true risk:

$$\lim_{n \to \infty} P\left(Dis(\hat{\theta}_n^{ERM}) \geq 0\right) = 1/2, \tag{12}$$

One way to interpret the *Optimizer's Curse* is that by using argmax on the estimates of expected reward (optimized via ERM) **we are ignoring their variance/uncertainty levels**, and **overfitting** the overall decision to the training sample.

The Optimizer's Curse in Recommendation

- The *Optimizer's Curse* mainly appears when the **reward estimates are based on small sample sizes** and **the deltas between the true mean rewards** of the different actions **are relatively small** such that the variance of the empirical estimators can lead to overoptimistic expectations or even erroneous ranking of actions.
- **This is not a corner case!** In real recommendation applications, the space of possible context-action pairs is extremely large, so there are always areas of the distribution where the current system is deciding based on very low sample sizes.

Fixing Covariate Shift and Optimizer's Curse with Policy Learning

We saw that value-based approaches for decision-making suffer from two issues **when using ERM-based models**.

In the following sub-sections we will address in turn the two issues and show how Policy Learning / Off-Policy Contextual Bandits can help.

Section 3: Fixing ERM issues using Policy Learning

3.1. Fixing Covariate Shift using Counterfactual Risk Minimization

Moving away from ERM

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 **Value-based Methods**

Recommendation Application: Causal Product Embeddings

- The work in [Bonner and Vasile, 2018] proposes a joint matrix factorization technique that solves the Covariate Shift Problem using the IPS trick along these lines.
- The paper proposes an algorithm that is able to leverage a small exploration sample of feedback collected under uniform exposure π_{unif} and a large sample of biased logged data π_0, in order to create action embeddings that are optimized for prediction under uniform exposure.
- The method shows a significant uplift versus the classical ERM-based solution.

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**:

$$R^{CRM}(\theta) = \mathbb{E}_{x\sim\nu, y\sim\pi_\theta(x)}\left[c(x,y)\right] = \mathbb{E}_{x\sim\nu, y\sim\pi_0(x)}\left[c(x,y)\frac{\pi_\theta(y|x)}{\pi_0(y|x)}\right], \quad (13)$$

Counterfactual Risk Minimization (CRM)

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

$$\hat{\theta}_n^{CRM} \triangleq \underset{\theta \in \Theta}{\operatorname{argmin}} \hat{R}_n^{CRM}(\theta) \tag{14}$$

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

Note: Off-Policy = Counterfactual

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.

Practical Session: VanillaCB

VanillaCB: 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

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

VanillaCB. Formulation

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

CRM Objective:

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

This objective is not straightforward to optimize, but we can lower bound its log using Jensen inequality

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

VanillaCB. Formulation

$\hat{\theta}$ 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).

VanillaCB. Exercise

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.

VanillaCB. 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 of bandits and RL.

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.

Well, it turns out 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:

$$\lim_{n \to \infty} P\left(Dis(\hat{\theta}_n) \geq 0\right) \leq \delta, \tag{15}$$

Distributionally Robust Risk

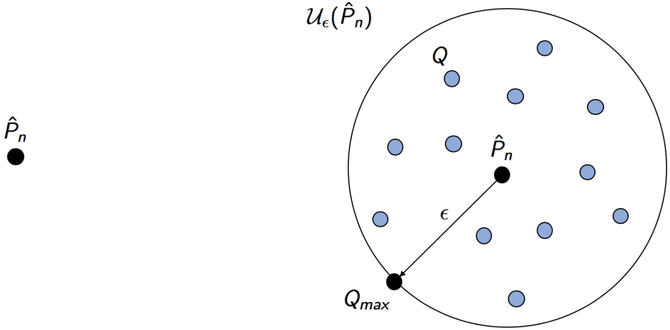
Recall that the source of the Optimizer's curse is that at decision time, we do not take into account the uncertainty level of the reward/risk estimator, but only its expectation (which ERM optimizes for).

One way to circumvent the ERM limitations is to treat the empirical training distribution $\hat{P}_n$ over (state,action) pairs with *skepticism* and to replace it with an uncertainty set $\mathcal{U}_\epsilon(\hat{P}_n)$ of distributions around $\hat{P}_n$ that are *consistent* with the data (where $\epsilon > 0$ is a parameter controlling the size of the uncertainty set $\mathcal{U}_\epsilon(\hat{P}_n)$).

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

$$\hat{\mathbf{R}}_n^{\mathcal{U}}(\theta) \triangleq \max_{Q \in \mathcal{U}_\epsilon(\hat{P}_n)} \mathbb{E}_{(x,y)\sim Q}[\ell(x, y; \theta)]. \tag{16}$$

Distributionally Robust Risk



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

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

Figure 7: Empirical Risk vs. Empirical Robust Risk

Distributional Robust Optimization (DRO)

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

$$\hat{\theta}_n^{\text{rob}} \triangleq \text{argmin}_{\theta\in\Theta}\hat{\mathbf{R}}_n^{\mathcal{U}}(\theta) = \underset{\theta\in\Theta}{\text{argmin}} \max_{Q\in\mathcal{U}_\epsilon(P_n)} \mathbb{E}_{(x,y)\sim Q}[\ell(x,y;\theta)]. \quad (17)$$

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 $\mathcal{U}_\epsilon(\hat{P}_n)$ defined using *coherent φ-divergences*, one can prove asymptotic performance guarantees - see [Duchi et al., 2016]:

$$\lim_{n\to\infty} P\left(Dis(\hat{\theta}_n^{DRO}) \geq 0\right) \leq \delta, \tag{18}$$

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

$$\lim_{n\to\infty} P\left(Dis(\hat{\theta}_n^{ERM}) \geq 0\right) = 1/2, \tag{19}$$

The DRO solution is asymptotically consistent

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

$$\lim_{n \to \infty} Dis(\hat{\theta}_n^{DRO}) = 0 \tag{20}$$

$$\lim_{n \to \infty} Reg(\hat{\theta}_n^{DRO}) = 0 \tag{21}$$

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:

$$\hat{\theta}_n^{\text{CRM-rob}} \triangleq \underset{\theta \in \Theta}{\operatorname{argmin}} \max_{Q \in \mathcal{U}_\epsilon(P_n)} \mathbb{E}_{(x,y)\sim Q}[\ell^{CRM}_{(x,y;\theta)}]. \tag{22}$$

DRO-CRM with coherent φ-divergences

The minimization of the robust risk $\hat{\mathbf{R}}_n^{\varphi}(\theta)$ based on coherent information divergences is *asymptotically equivalent* with a variance-sensitive policies selection process, that is penalizing policies that are *far* from the logging policy (as measured by the empirical risk variance) - see [Duchi et al., 2016]

This action selection process is also known as **Sample Variance Penalization (SVP)**

DRO-CRM $=$ SVP

More formally,

DRO-CRM is asymptotically equivalent with Sample Variance Penalization:

$$\hat{\mathbf{R}}_n^{\varphi}(\theta) = \hat{R}_n(\theta) + \sqrt{\epsilon V_n(\theta)} + \alpha_n(\theta), \tag{23}$$

where:

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

DRO-CRM Methods

So far, there are two published methods for DRO-CRM:

- **DRO**-χ^2 which has the exact SVP objective, but the optimization introduces an approximation - see: [Swaminathan and Joachims, 2015]
- **DRO-KL** which approximates the SVP objective, but results in an easier optimization task - see: [Faury et al., 2019]

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**.

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).

Practical Session: OrganicCB

OrganicCB: Intuition

Translate data in a more meaningful space where the logged policy π_0 and the learned policy π_θ are closer.

This leads to less variance in IPS and thus less disappointment.

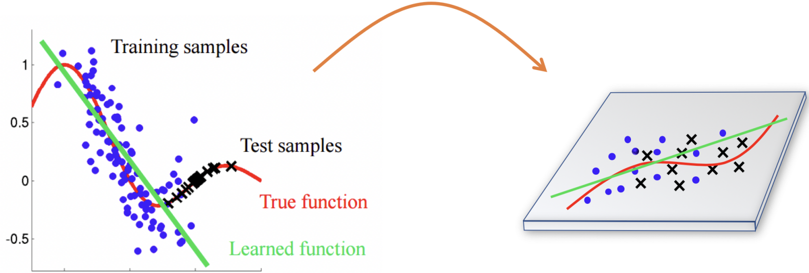


Figure 8: Train samples (blue) and test samples (black) in the original space (left) and the embedded space (right).

OrganicCB: A Simple model

Goal : avoid overfitting on few observations that would lead to a highly optimized CRM objective but with very high variance.

Mean : Translate all observations into an embedded space where observations are more related to each other, for example with **SVD product embeddings** based on organic data.

One solution : build the user state as an average mean of the seen products. Namely,

$$s_i = \frac{1}{\sum_j \mathrm{seen}(i, p_j)} \sum_j \mathrm{emb}(p_j) \cdot \mathrm{seen}(i, p_j),$$

where $\mathrm{emb}(p_j)$ is the embedding of the product j and $\mathrm{seen}(i, p_j)$ is the number of times the user i has seen the product p_j.

OrganicCB: Exercice

Create a new agent that works like VanillaCB but which takes the average mean of the seen products embeddings for the user state.

This solution is one possibility among many. You might also try to:

- First define the user state as the average mean of the seen products embeddings and then take as features the similarities of this user state with each product embedding.
- Use a neural network to learn the best representation of the user state.

OrganicCB: Conclusions

Using organic user embeddings is valuable to fight the optimizer curse for two reasons:

- It uses organic data that is not affected by our policy, leading to less variance in IPS.
- It shares knowledge between products and allows to do transfer learning to avoid overestimating one specific action.

In both cases, this becomes even more valuable when the number of products increases.

Section 4. Recap and Conclusions

Organic and Bandit tasks. A Recap of Methods

Feedback	Framework	Task	Question
Organic	Sequence prediction	Predict $Proba(P_+ \| U)$	What is the proba of P being observed next (+), given U?
Organic	Classification	Predict $Proba(+ \| U, P)$	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 $Proba(Click \| U, A)$	Predict probaClick for all (U,A) pairs
Bandit	Policy Search	Predict $Proba(A \| U) \times \frac{Click}{\pi_0}$	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 $Proba(A\|U) \times \frac{Click}{\pi_0}$	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 $Proba(A\|U) \times \frac{Click}{\pi_0}$	W-Multinomial LogReg

Future Reco: DRO-CRM + Organic Transfer

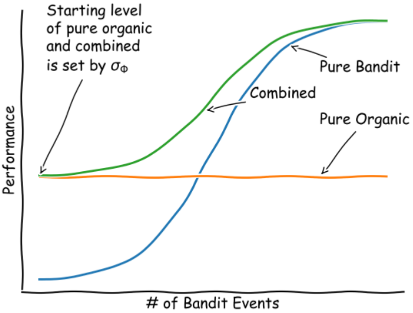


Figure 9: Joint Policy Learning: Combine Organic and Bandit feedback

Finally, a shameless plug!

You like Recommender Systems? You want to understand how to evaluate them offline and to avoid all of the problems above?

Attend / Submit at our **REVEAL workshop at RecSys 2019** in Copenhagen: `https://recsys.acm.org/recsys19/reveal/`

Thank You!

Questions?

- Bonner, S. and Vasile, F. (2018).
  Causal embeddings for recommendation.
  In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 104–112. ACM.

- 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.

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