Logged Bandits

Onur Atan
What is logged data?

In the Multi-Armed Bandit problem:

- Learner receives contextual information $x_n$
- Learner takes the action $a_n$
- Learner receives the feedback $Y_{n\cdot a_n}$

Logged data is the collection of these triplets $(x_n, a_n, Y_{n\cdot a_n})$
Neyman-Rubin Causal Model

- Each patient $i$ is associated with feature vector $X_i$
- Treatment alternatives: $\mathcal{A} = \{0, 1, \ldots, k - 1\}$
- Potential outcomes: $Y^0, Y^1, \ldots, Y^{k-1}$
- Observational data: $\mathcal{D}^N = \{X_i, A_i, Y^A_i\}$
- Personalized treatment policy: $h : \mathcal{X} \rightarrow \mathcal{A}$
- Policy value: $V(h) = \mathbb{E}_X \left[ Y^{h(X)} \right]$
Main Assumptions

- **Unconfoundedness**: Potential outcomes are independent of the treatment performed given the feature vector: $Y^0, Y^1, \ldots, Y^k \perp A \mid X$.

- **Overlap**: There is a non-zero probability that each patient receiving different treatment alternatives.

These assumptions allow us to infer the outcomes of counterfactual actions.
Difference between Supervised and Off-Policy Learning

- Only the outcome of the treatment actually performed is observed: Partial label
- Treatments are selected by clinicians (experts) based on features: Selection bias
- Example: Simpson’s Paradox

<table>
<thead>
<tr>
<th></th>
<th>Overall</th>
<th>Small stones</th>
<th>Large stones</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Surgery</td>
<td>78% (273/350)</td>
<td>93% (81/87)</td>
<td>73% (192/263)</td>
</tr>
<tr>
<td>Percutaneous Nephrolithotony</td>
<td>83% (289/350)</td>
<td>87% (234/270)</td>
<td>69% (55/80)</td>
</tr>
</tbody>
</table>

Table. Success rate of treatments on large and small stone patients

- Large feature and action space
- The interactions between features and treatments are not known
Main Objectives: ITE Estimation, Policy Optimization (PO)

Two different objectives: ITE estimation, Policy Optimization (PO)

ITE problem: estimate the expected difference between treatment and control outcomes given feature vector.

Policy Optimization (PO): find a policy mapping features to actions that maximizes the expected outcomes.

PO is easier than ITE – one can turn ITE into action recommendation but not the other way around. But ITE literature mostly focuses on problems with 2 treatment alternatives.
Importance Sampling (IS) Estimator

- Importance Sampling Estimator is unbiased. Assuming data is collected with respect to \( \pi_0 \)

\[
\hat{V} (\pi) = \frac{1}{N} \sum_{n=1}^{N} \frac{Y_n^{a_n} \pi(a_n | x_{-n})}{\pi_0(a_n | x_n)}
\]

- Importance Sampling Estimator has large variance

- Different estimators: self-normalizing, doubly robust
Linear Policy: $\pi(a | x) = \frac{\exp(W_a x)}{\sum_a \exp(W_a x)}$

Maximize the IS estimator minus the variance of the estimator

$$\max_{\pi} \hat{V}^{1S}(\pi) - \lambda \sqrt{\text{var}(\hat{V}^{1S}(\pi))}$$
## Related Work

<table>
<thead>
<tr>
<th>Literature</th>
<th>Propensities known</th>
<th>Objective</th>
<th>Actions</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shalit (2017)</td>
<td>NO</td>
<td>ITE</td>
<td>2</td>
<td>Representation balancing</td>
</tr>
<tr>
<td>Alaa &amp; Schaar (2017)</td>
<td>NO</td>
<td>ITE</td>
<td>2</td>
<td>Risk based empirical Bayes</td>
</tr>
<tr>
<td>Swaminathan (2015)</td>
<td>YES</td>
<td>PO</td>
<td>&gt; 2</td>
<td>IPS re-weighting</td>
</tr>
<tr>
<td>Ours</td>
<td>NO</td>
<td>PO</td>
<td>&gt; 2</td>
<td>Representation balancing</td>
</tr>
</tbody>
</table>

- Our work is different than ITE/CATE estimation because:
  - Ours is learning probabilities over actions (policy) to learn best actions
  - We have NO restrictions on number of actions

- Our work is different than existing work in Policy Optimization because:
  - NO knowledge about logging policy is assumed to be known.
  - Swaminathan (2015) uses the inverse propensities to handle the bias, but ours uses domain adaption to handle the bias.
Deep-Treat (1)

Figure 1: Neural network Model
The bias-removing neural networks has two components: reconstruction loss and cross-entropy loss between $\widehat{Pr}(A)$ and $\widehat{Pr}(A|\Phi(X))$.

The cross-entropy loss:

$$\ell_{ce} \left( \widehat{Pr}(A), \widehat{Pr}(A|\Phi(x)) \right) =$$

$$- \sum_a \widehat{Pr}(A)(\theta_t \Phi_n - \log(\sum_a \exp(\theta_t \Phi_n)))$$

Where $\theta_t$ is the parameter of logistic regression model fitted.
Policy Optimization as a Transfer Learning

- **Representation function**: \( \Phi : \mathcal{X} \rightarrow \mathcal{Z} \)
- **Hypothesis class**: \( \mathcal{H} \)
- **Source distribution**: \( \mathcal{D}^\Phi_S \) on the samples \((X, A)\) in observational data
- **Target distribution**: \( \mathcal{D}^\Phi_T \) on the samples \((X, Q)\) where \(X\) follows the same marginal distribution in observational data, \(Q\) is generated independently from multinomial distribution with probabilities \(1/k\).
- **Source and target value functions**: for \( I \in \{S, T\} \)
  \[
  V^\Phi_I(h) = k \mathbb{E}_{(Z, A) \sim \mathcal{D}^\Phi_I} \left[ Y^{h(Z)} 1(h(Z) = A) \right]
  \]
Counterfactual Estimation Bounds

- Target value is unbiased: \( \hat{V}_T(h) = V(h) \)

- **The bias of the source value**: \( |\hat{V}_S^\Phi(h) - V^\Phi(h)| \leq kd_H(\mathcal{D}_S^\Phi, \mathcal{D}_T^\Phi) \) where \( d_H \) is H-divergence between source and target.

- **Monte-Carlo estimator**: \( \hat{V}_S(h) = \frac{k}{n} \sum_{i=1}^{n} A_i \mathbb{1}(h(X_i) = A_i) \)

- **The estimation bound**: \( |\hat{V}_S^\Phi(h) - V^\Phi(h)| \leq d_H(\hat{\mathcal{D}}_S^\Phi, \hat{\mathcal{D}}_T^\Phi) + \mathcal{O}\left( \sqrt{\frac{N_\infty(1/n, H, 2n)}{n}} \right) \)

- **Counterfactual Policy Optimization**: maximize \( \phi, h \) \( \hat{V}_S^\Phi(h) - \lambda_0 d_H(\hat{\mathcal{D}}_S^\Phi, \hat{\mathcal{D}}_T^\Phi) - \lambda_1 \| h \| \)
Figure. Domain Adversarial Neural Network model based on [Ganin, 2016]
DACPOL Components

\[ \widehat{\mathcal{D}}_S = \{(X_i, A_i, Y_i^A) : i = 1, 2, \ldots, n\} \]

where \( Q_i \sim \text{Multinomial}([1/k, \frac{1}{4}, 1/k]) \).

\[ \widehat{\mathcal{D}}_T = \{(X_i, Q_i) : i = 1, 2, \ldots, n\} \]

**Input:**

- **Representation layer:** maps features to representations
- **Policy layer:** maps representations to policy
- **Domain layer:** maps representations to probability of data being from target.
- **Reversal layer:** reverse gradients of domain loss in backward propagation in order to learn representations that are indifferent between source and target.