Batch Learning from Bandit Feedback

(Adith Swaminathan), Thorsten Joachims

Department of Computer Science
Department of Information Science
Cornell University


Batch Learning from Bandit Feedback

- Data
  \[ S = \{(x_1, y_1, \delta_1), \ldots, (x_n, y_n, \delta_n)\} \]
  \(\rightarrow\) Partial Information (aka “Bandit”) Feedback

- Properties
  - Contexts \(x_i\) drawn i.i.d. from unknown \(P(X)\)
  - Actions \(y_i\) selected by existing system \(\pi_0: X \rightarrow Y\)
  - Feedback \(\delta_i\) drawn i.i.d. from unknown \(\delta: X \times Y \rightarrow \mathbb{R}\)

- Goal of Learning
  - Find new system \(\pi\) that selects \(y\) with better \(\delta\)

Historic Interaction Logs: News Recommender

- Context \(x\):
  - User
- Action \(y\):
  - Portfolio of news articles
- Feedback \(\delta(x, y)\):
  - Reading time in minutes

Historic Interaction Logs: Ad Placement

- Context \(x\):
  - User and page
- Action \(y\):
  - Ad that is placed
- Feedback \(\delta(x, y)\):
  - Click / no-click

Historic Interaction Logs: Search Engine

- Context \(x\):
  - Query
- Action \(y\):
  - Ranking
- Feedback \(\delta(x, y)\):
  - win/loss against baseline in interleaving

Comparison with Supervised Learning

<table>
<thead>
<tr>
<th></th>
<th>Batch Learning from Bandit Feedback</th>
<th>Full-Information Supervised Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train example</td>
<td>((x, y, \delta))</td>
<td>((x, y^*))</td>
</tr>
<tr>
<td>Context (x)</td>
<td>drawn i.i.d. from unknown (P(X))</td>
<td>drawn i.i.d. from unknown (P(X))</td>
</tr>
<tr>
<td>Action (y)</td>
<td>selected by existing system (\pi_0: X \rightarrow Y)</td>
<td>N/A</td>
</tr>
<tr>
<td>Feedback (\delta)</td>
<td>Observe (\delta(x, y)) only for (y) chosen by (\pi_0)</td>
<td>Assume known loss function (\Delta(y, y^*))</td>
</tr>
</tbody>
</table>
Learning Settings

<table>
<thead>
<tr>
<th>Online Learning</th>
<th>Batch Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Perceptron</td>
<td>• SVM</td>
</tr>
<tr>
<td>• Winnow</td>
<td>• Random Forests</td>
</tr>
<tr>
<td>• Etc.</td>
<td>• Etc.</td>
</tr>
<tr>
<td>• EXP3</td>
<td>• Offset Tree</td>
</tr>
<tr>
<td>• UCB1</td>
<td>• (Off-Policy RL)</td>
</tr>
</tbody>
</table>

Outline of Lecture

- Batch Learning from Bandit Feedback (BLBF) \( S = \{(x_1, y_1, \delta_1), \ldots, (x_n, y_n, \delta_n)\} \)
  - Find new system \( \pi \) that selects \( y \) with better \( \delta \)
- Learning Principle for BLBF
  - Hypothesis Space, Risk, Empirical Risk, and Overfitting
  - Counterfactual Risk Minimization
- Learning Algorithm for BLBF
  - POEM for Structured Output Prediction
- Improved Counterfactual Risk Estimators
  - Self-Normalizing Estimator

Hypothesis Space

Definition [Stochastic Hypothesis / Policy]:
Given context \( x \), hypothesis/policy \( \pi \) selects action \( y \) with probability \( \pi(y|x) \)

\[
\pi_1(y|x) \quad \pi_2(y|x)
\]

Risk

Definition [Expected Loss (i.e. Risk)]:
The expected loss / risk \( R(\pi) \) of policy \( \pi \) is

\[
R(\pi) = \int \int \delta(x, y) \pi(y|x) P(x) \, dx \, dy
\]

On-Policy Risk Estimation

Given \( S = (x_1, y_1, \delta_1), \ldots, (x_n, y_n, \delta_n) \) collected under \( h_0 \),

\[
\hat{R}(\pi_0) = \frac{1}{n} \sum_{i=1}^{n} \delta_i
\]

\( \Rightarrow \) A/B Testing
Field \( h_1 \): Draw \( x \sim P(x) \), predict \( y \sim \pi_1(y|x) \), get \( \delta(x, y) \)
Field \( h_2 \): Draw \( x \sim P(x) \), predict \( y \sim \pi_2(y|x) \), get \( \delta(x, y) \)
Field \( h_{[\pi]} \): Draw \( x \sim P(x) \), predict \( y \sim \pi_{[\pi]}(y|x) \), get \( \delta(x, y) \)

Approach 1: Model the World

- Approach [Athey & Imbens, 2015] for \( Y = \{y_0, y_1\} \):
  - Learning: estimate CATE \( E[\delta(x, y_1) - \delta(x, y_0)|x] \) via regression
  - For \( f(x) \) from \( x \) to \( -\delta(x, y_1)/p_i \) if \( y_i = y_0 \)
  - \( \delta(x, y_0)/p_i \) otherwise
- New policy: Given \( x \), select \( y = \begin{cases} y_0 & \text{if } f(x) < 0 \\ y_1 & \text{otherwise} \end{cases} \)

\( \Rightarrow \) More general: “reward simulator approach”, “model-based reinforcement learning”, …
Approach 2: Model the Selection Bias

Given $S = \{(x_1, y_1, \delta_1), ..., (x_n, y_n, \delta_n)\}$ collected under $\pi_{\Theta}$,

$$\hat{R}(\pi) = \frac{1}{n} \sum_{i=1}^{n} \frac{\pi(y_i|x_i)}{\pi_{\Theta}(y_i|x_i)} \delta_i$$

→ Get unbiased estimate of risk, if propensity nonzero everywhere (where it matters).

Generalization Error Bound for BLBF

- Theorem [Generalization Error Bound]
  - For any hypothesis space $H$ with capacity $C$, and for all $\pi \in H$ with probability $1 - \eta$
  $$R(\pi) \leq \hat{R}(\pi) + O\left(\sqrt{\frac{\text{Var}(\pi)}{n}}\right) + O(C)$$
  $$\hat{R}(h) = \text{Mean} \left( \frac{\pi(y_i|x_i)}{p_i} \delta_i \right)$$
  $$\text{Var}(h) = \text{Var} \left( \frac{\pi(y_i|x_i)}{p_i} \delta_i \right)$$

→ Bound accounts for the fact that variance of risk estimator can vary greatly between different $\pi \in H$

Outline of Lecture

- Batch Learning from Bandit Feedback (BLBF)
  $S = \{(x_1, y_1, \delta_1, p_1), ..., (x_n, y_n, \delta_n, p_n)\}$
  → Find new system $h$ that selects $y$ with better $\delta$

- Learning Principle for BLBF
  - Hypothesis Space, Risk, Empirical Risk, and Overfitting
  - Counterfactual Risk Minimization
  → Learning Algorithm for BLBF
  - POEM for Structured Output Prediction
  - Improved Counterfactual Risk Estimators
    - Self-Normalizing Estimator

Partial Information

Empirical Risk Minimization

- Training $\hat{h} := \arg\min_{\pi \in H} \frac{n}{\delta_i}$

Counterfactual Risk Minimization

- Theorem [Generalization Error Bound]
  $$R(\pi) \leq \hat{R}(\pi) + O\left(\sqrt{\frac{\text{Var}(\pi)}{n}}\right) + O(C)$$

→ Constructive principle for designing learning algorithms

$$\pi_{\text{erm}} = \arg\min_{\pi \in H_1} \hat{R}(\pi) + \lambda_1 \left(\sqrt{\text{Var}(\pi)}\right) + \lambda_2 C(H_1)$$

POEM Hypothesis Space

Hypothesis Space: Stochastic prediction rules

$$\pi(y|x, w) = \frac{1}{Z(x)} \exp(w \cdot \Phi(x, y))$$

with
- $w$: parameter vector to be learned
- $\Phi(x, y)$: joint feature map between input and output
- $Z(x)$: partition function

Note: same form as CRF or Structural SVM
POEM Learning Method

- Policy Optimizer for Exponential Models (POEM)
  - Data: \( S = \{ (x_1, y_1, \delta_1, p_1), \ldots, (x_n, y_n, \delta_n, p_n) \} \)
  - Hypothesis space: \( \pi(y|x, w) = \exp(w \cdot \phi(x, y))/Z(x) \)
  - Training objective: Let \( z_i(w) = \pi(y_i|x_i, w)\delta_i/p_i \)

\[
w = \arg\min_{w\in\mathbb{R}^d} \left[ \frac{1}{n} \sum_{i=1}^{n} z_i(w) + \lambda_1 \left( \frac{1}{n} \sum_{i=1}^{n} z_i(w)^2 \right) \right] + \lambda_2 |w|^2
\]

- Unbiased Risk Estimator
- Variance Control
- Capacity Control

POEM Experiment

Multi-Label Text Classification

- Data: \( S = \{ (x_1, y_1, \delta_1, p_1), \ldots, (x_n, y_n, \delta_n, p_n) \} \)
  - \( x \): Text document
  - \( y \): Predicted label vector
  - \( \delta \): number of incorrect labels in \( y \)
  - \( p_i \): propensity under logging policy \( h_i \)

- Results: Reuters LYRL RCV1 (top 4 categories)

POEM Efficient Training Algorithm

- Training Objective:

\[
OPT = \min_{w\in\mathbb{R}^d} \left[ \frac{1}{n} \sum_{i=1}^{n} z_i(w) + \lambda_1 \left( \frac{1}{n} \sum_{i=1}^{n} z_i(w)^2 \right) \right] + \lambda_2 |w|^2
\]

- Idea: First-order Taylor Majorization
  - Majorize \( \sqrt{z_i(w)} \) at current value
  - Majorize \( -z_i(w)^2 \) at current value

\[
OPT \leq \min_{w\in\mathbb{R}^d} \left[ \sum_{i=1}^{n} \lambda_1 z_i(w) + B_i z_i(w)^2 \right]
\]

- Algorithm:
  - Majorize objective at current \( w \)
  - Solve majorizing objective via Adagrad to get \( w_{t+1} \)

Is Variance Regularization Improve Generalization?

- IPS: \( w = \arg\min_{w\in\mathbb{R}^d} \left[ \frac{1}{n} \sum_{i=1}^{n} z_i(w)^2 \right] \)
- POEM: \( w = \arg\min_{w\in\mathbb{R}^d} \left[ \frac{1}{n} \sum_{i=1}^{n} z_i(w)^2 \right] + \lambda_1 \frac{1}{n} \sum_{i=1}^{n} z_i(w)^2 + \lambda_2 |w|^2 \)

- Hamming Loss
- Scene
- Yeast
- TMC
- LYRL

<table>
<thead>
<tr>
<th></th>
<th>Scene</th>
<th>Yeast</th>
<th>TMC</th>
<th>LYRL</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_0 )</td>
<td>1.543</td>
<td>5.547</td>
<td>3.445</td>
<td>1.463</td>
</tr>
<tr>
<td>IPS</td>
<td>1.519</td>
<td>4.614</td>
<td>3.023</td>
<td>1.118</td>
</tr>
<tr>
<td>POEM</td>
<td>1.143</td>
<td>4.517</td>
<td>2.522</td>
<td>0.996</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th># examples</th>
<th># features</th>
<th># labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hamming Loss</td>
<td>4*1211</td>
<td>30438</td>
<td>22</td>
</tr>
<tr>
<td>IPS</td>
<td>4*1500</td>
<td>47236</td>
<td>4</td>
</tr>
<tr>
<td>POEM</td>
<td>4*23149</td>
<td>4*23149</td>
<td>4</td>
</tr>
</tbody>
</table>

How computationally efficient is POEM?

- Batch Learning from Bandit Feedback (BLBF)
  - \( S = \{ (x_1, y_1, \delta_1, p_1), \ldots, (x_n, y_n, \delta_n, p_n) \} \)
  - Find new system \( h \) that selects \( y \) with better \( \delta \)

Outline of Lecture

- Batch Learning from Bandit Feedback (BLBF)
  - Hypothesis Space, Risk, Empirical Risk, and Overfitting
  - Counterfactual Risk Minimization
- Learning Principle for BLBF
  - POEM for Structured Output Prediction
- Improved Counterfactual Risk Estimators
  - Self-Normalizing Estimator

CPU Seconds

<table>
<thead>
<tr>
<th></th>
<th>Scene</th>
<th>Yeast</th>
<th>TMC</th>
<th>LYRL</th>
</tr>
</thead>
<tbody>
<tr>
<td>POEM</td>
<td>4.71</td>
<td>5.02</td>
<td>276.13</td>
<td>120.09</td>
</tr>
<tr>
<td>IPS</td>
<td>1.65</td>
<td>2.86</td>
<td>98.92</td>
<td>13.66</td>
</tr>
<tr>
<td>CRF (L-BFGS)</td>
<td>4.86</td>
<td>3.28</td>
<td>99.18</td>
<td>62.93</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th># examples</th>
<th># features</th>
<th># labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Seconds</td>
<td>4*1211</td>
<td>30438</td>
<td>22</td>
</tr>
<tr>
<td>IPS</td>
<td>4*1500</td>
<td>47236</td>
<td>4</td>
</tr>
<tr>
<td>POEM</td>
<td>4*23149</td>
<td>4*23149</td>
<td>4</td>
</tr>
</tbody>
</table>
### Counterfactual Risk Minimization

- **Theorem [Generalization Error Bound]**
  \[
  R(\pi) \leq \tilde{R}(\pi) + O \left( \sqrt{\text{Var}(\pi)/n} \right) + O(C)
  \]

  ⇒ Constructive principle for designing learning algorithms

  \[
  \pi_{\text{ERM}} = \arg\min_{\pi \in H_1} \tilde{R}(\pi) + \lambda_1 \left( \sqrt{\text{Var}(\pi)/n} \right) + \lambda_2 C(H_1)
  \]

  \[
  \tilde{R}(\pi) = \frac{1}{n} \sum_{i=1}^{n} \frac{\pi(y_i|x_i)}{p_i} \delta_i 
  \]

  \[
  \text{Var}(\pi) = \frac{1}{n} \sum_{i=1}^{n} \left( \frac{\pi(y_i|x_i)}{p_i} \delta_i \right)^2 - \tilde{R}(\pi)^2
  \]

  \[\text{[Swaminathan & Joachims, 2015]}\]

### Propensity Overfitting Problem

- **Example**
  - Instance Space \( X = \{1, \ldots, k\} \)
  - Label Space \( Y = \{1, \ldots, k\} \)
  - Loss \( \delta(x, y) = \begin{cases} 0 & \text{if } y = x \\ 1 & \text{otherwise} \end{cases} \)
  - Training data: uniform \( x, y \) sample
  - Hypothesis space: all deterministic functions
    \[
    \pi_{\text{opt}}(x) = x \text{ with risk } R(\pi_{\text{opt}}) = -2
    \]

  \[
  R(\tilde{R}) = \min_{\pi \in H_1} \frac{1}{n} \sum_{i=1}^{n} \frac{\pi(y_i|x_i)}{p_i} \delta_i = 1
  \]

  ⇒ Problem 1: Unbounded risk estimate!

- **Control Variates**
  - Idea: Inform estimate when expectation of correlated random variable is known.
    - Estimator:
      \[
      R(\tilde{R}) = \frac{1}{n} \sum_{i=1}^{n} \frac{\pi(y_i|x_i)}{p_i} \delta_i
      \]
    - Correlated RV with known expectation:
      \[
      S(\tilde{R}) = \frac{1}{n} \sum_{i=1}^{n} \frac{\pi(y_i|x_i)}{p_i} \delta_i
      \]
      \[
      E[S(\tilde{R})] = \frac{1}{n} \sum_{i=1}^{n} \frac{\pi(y_i|x_i)}{p_i} \delta_i P(x_i)dy_i dx_i = 1
      \]

  ⇒ New Risk Estimator: Self-normalizing estimator
    \[
    \tilde{R}(\pi) = \frac{\tilde{R}(\pi)}{S(\tilde{R})}
    \]

### Norm-POEM Learning Method

- **Method:**
  - Data: \( S = \{(x_1, y_1, \delta_1, p_1), \ldots, (x_n, y_n, \delta_n, p_n)\} \)
  - Hypothesis space: \( \pi(x, y, w) = \exp(w \cdot \phi(x, y))/Z(x) \)
  - Training objective: Let \( R_2(w) = \pi(y_i|x_i, w)\delta_i/p_i \)

  \[
  w = \arg\min_{w \in \mathbb{R}^M} \left[ \tilde{R}(w) + \lambda_1 \sqrt{\text{Var}(\tilde{R}(w))} + \lambda_2 \|w\|^2 \right]
  \]

  \[\text{[Swaminathan & Joachims, 2015]}\]

### How well does Norm-POEM generalize?

<table>
<thead>
<tr>
<th>Loss</th>
<th>Scene</th>
<th>Yeast</th>
<th>TMC</th>
<th>LYRL</th>
</tr>
</thead>
<tbody>
<tr>
<td>(h_0)</td>
<td>1.511</td>
<td>5.577</td>
<td>3.442</td>
<td>1.459</td>
</tr>
<tr>
<td>POEM</td>
<td>1.200</td>
<td>4.520</td>
<td>2.152</td>
<td>0.914</td>
</tr>
<tr>
<td>Norm-POEM</td>
<td>1.045</td>
<td>3.876</td>
<td>2.072</td>
<td>0.799</td>
</tr>
</tbody>
</table>

- # examples: \(4 \times 1211\), \(4 \times 1500\), \(4 \times 21519\), \(4 \times 23149\)
- # features: 294, 103, 30438, 47236
- # labels: 6, 14, 22, 4
Conclusions

• Batch Learning from Bandit Feedback (BLBF)
  \[ S = (x_1, y_1, \delta_1, p_1), \ldots, (x_n, y_n, \delta_n, p_n) \]

• Learning Principle for BLBF
  \( \rightarrow \) Counterfactual Risk Minimization

• Learning Algorithm for BLBF
  \( \rightarrow \) POEM for Structured Output Prediction
  \( \rightarrow \) Efficient Training Method

• Open Questions
  – Counterfactual Risk Estimators
    \( \rightarrow \) Self-normalizing Estimator
  – Exploiting Smoothness in Loss Space
  – Exploiting Smoothness in Predictor Space
  – Propensity Estimation