Batch Learning from Bandit Feedback

CS7792 Counterfactual Machine Learning – Fall 2018

Thorsten Joachims

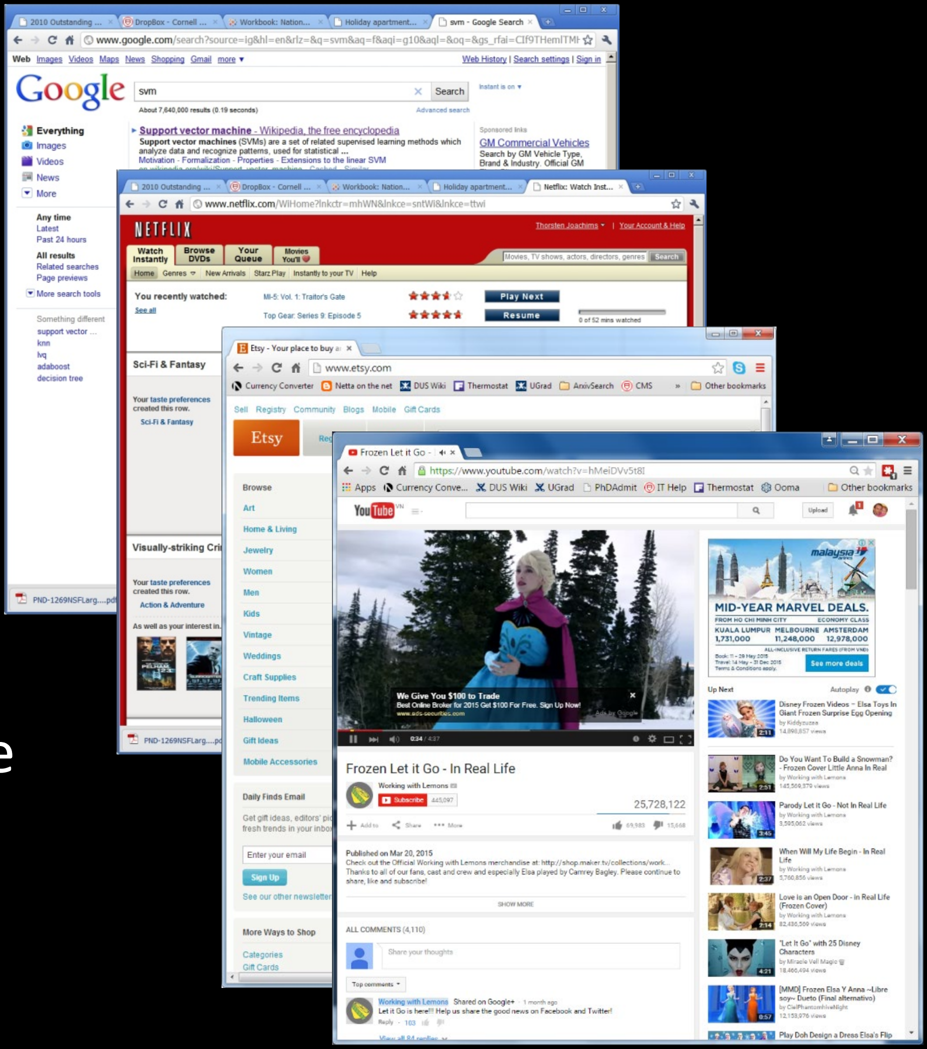
Departments of Computer Science and Information Science

Cornell University

- A. Swaminathan, T. Joachims, Batch Learning from Logged Bandit Feedback through Counterfactual Risk Minimization, JMLR Special Issue in Memory of Alexey Chervonenkis, 16(1):1731-1755, 2015.
- T. Joachims, A. Swaminathan, M. de Rijke. Deep Learning with Logged Bandit Feedback. In ICLR, 2018.

Interactive Systems

- Examples
  - Ad Placement
  - Search engines
  - Entertainment media
  - E-commerce
  - Smart homes
- Log Files
  - Measure and optimize performance
  - Gathering and maintenance of knowledge
  - Personalization



Batch Learning from Bandit Feedback

- Data

$$S = \big((x_1, y_1, \delta_1, p_1), \ldots, (x_n, y_n, \delta_n, p_n)\big)$$

context π_0 action reward propensity

→ "Bandit" Feedback

- Properties
  - Contexts x_i drawn i.i.d. from unknown $P(X)$
  - Actions y_i selected by existing system π_0: $X \rightarrow Y$
  - Loss δ_i drawn i.i.d. from unknown $P(\delta_i | x_i, y_i)$
- Goal of Learning
  - Find new system π that selects y with better δ

[Zadrozny et al., 2003] [Langford & Li], [Bottou, et al., 2014]

Learning Settings

	Full-Information (Labeled) Feedback	**Partial-Information (e.g. Bandit) Feedback**
Online Learning	• Perceptron • Winnow • Etc.	• EXP3 • UCB1 • Etc.
Batch Learning	• SVM • Random Forests • Etc.	?

Comparison with Supervised Learning

	Batch Learning from Bandit Feedback	**Conventional Supervised Learning**
Train example	(x, y, δ)	(x, y^*)
Context x	drawn i.i.d. from unknown $P(X)$	drawn i.i.d. from unknown $P(X)$
Action y	selected by existing system $h_0: X \rightarrow Y$	N/A
Feedback δ	Observe $\delta(x, y)$ only for y chosen by h_0	Assume known loss function $\Delta(y, y^*)$ → know feedback $\delta(x, y)$ for every possible y

Outline of Lecture

- Batch Learning from Bandit Feedback (BLBF)

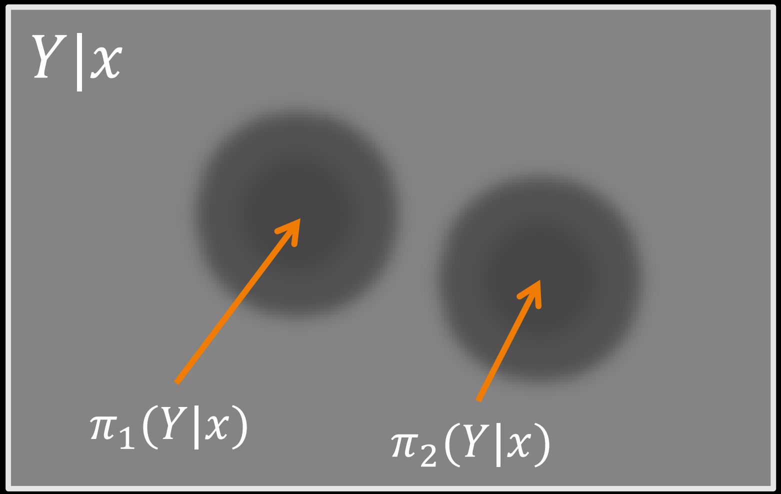
$$S = \big((x_1, y_1, \delta_1), \dots, (x_n, y_n, \delta_n)\big)$$

  → Find new policy π that selects y with better δ
- Learning Principle for BLBF
  - Hypothesis Space, Risk, Empirical Risk, and Overfitting
  - Learning Principle: Counterfactual Risk Minimization
- Learning Algorithms for BLBF
  - POEM: Bandit training of CRF policies for structured outputs
  - BanditNet: Bandit training of deep network policies

Hypothesis Space

Definition [Stochastic Hypothesis / Policy]:

Given context x, hypothesis/policy π selects action y with probability $\pi(y|x)$



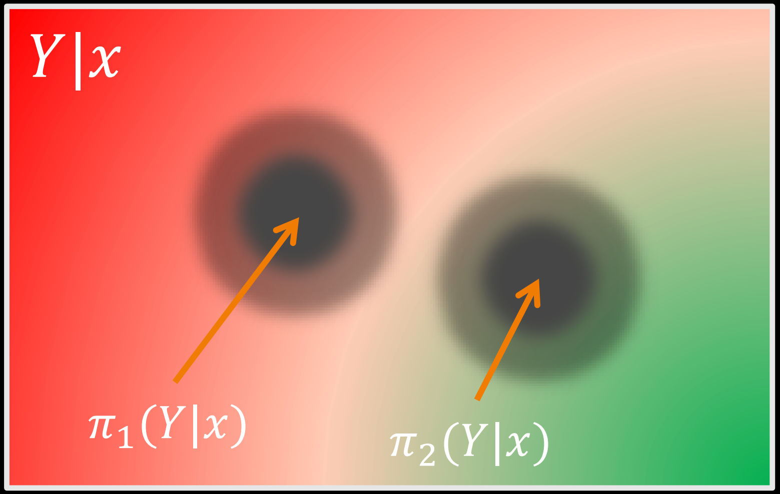
Note: stochastic prediction rules $\supset$ deterministic prediction rules

Risk

Definition [Expected Loss (i.e. Risk)]:

The expected loss / risk $\mathrm{R}(\pi)$ of policy π is

$$\mathrm{R}(\pi) = \int\int \delta(x,y)\pi(y|x)P(x)\, dx\, dy$$

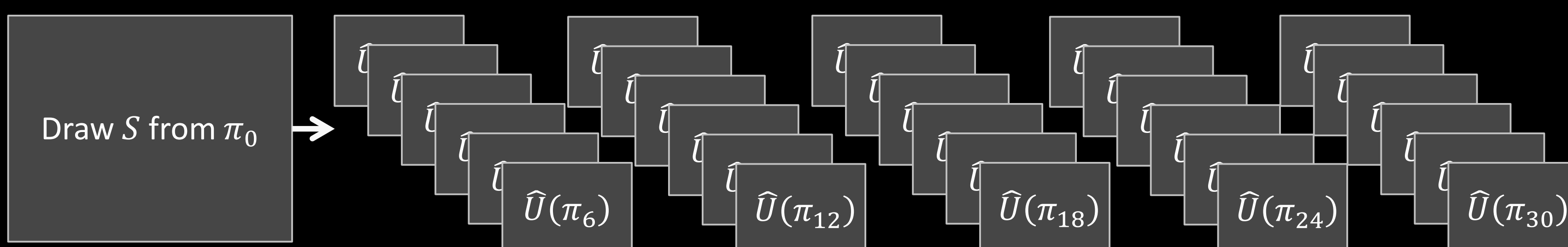


Evaluating Online Metrics Offline

- Online: On-policy A/B Test

Draw S_1 from π_1 → $\hat{U}(\pi_1)$	Draw S_2 from π_2 → $\hat{U}(\pi_2)$	Draw S_3 from π_3 → $\hat{U}(\pi_3)$	Draw S_4 from π_4 → $\hat{U}(\pi_4)$	Draw S_5 from π_5 → $\hat{U}(\pi_5)$	Draw S_6 from π_6 → $\hat{U}(\pi_6)$	Draw S_7 from π_7 → $\hat{U}(\pi_7)$

- Offline: Off-policy Counterfactual Estimates



Approach 1: Direct Method

- Data:
  $S = \big((x_1, y_1, \delta_1), \dots, (x_n, y_n, \delta_n)\big)$

1. Learn reward predictor

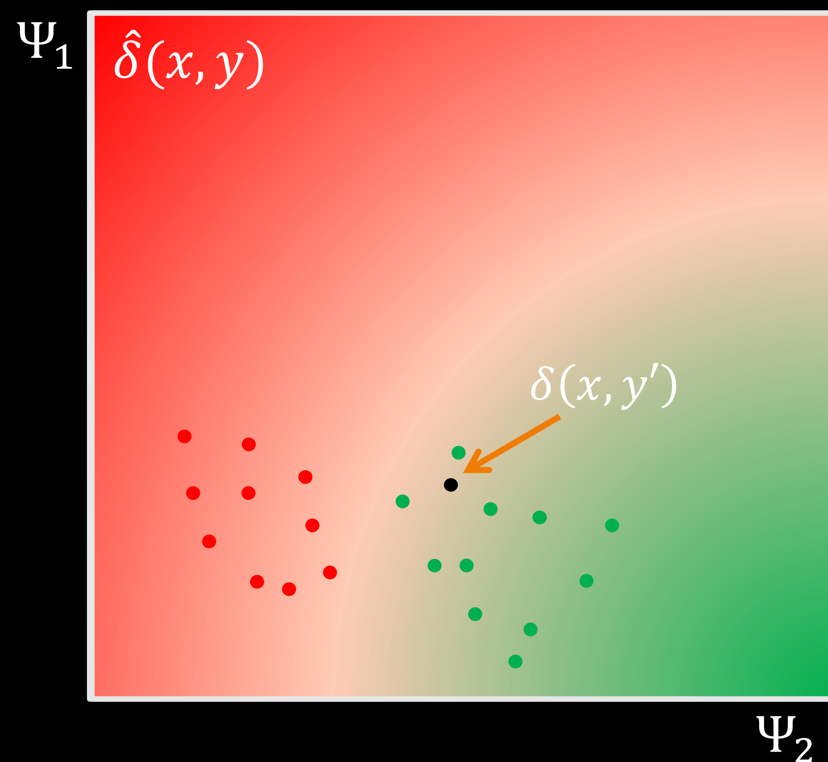
$$\hat{\delta}: x \times y \to \Re$$

   Represent via features $\Psi(x, y)$

   Learn regression based on $\Psi(x, y)$ from S collected under π_0

2. Derive policy $\pi(x)$

$$\pi(x) \stackrel{\text{def}}{=} \operatorname*{argmax}_{y} \left[\hat{\delta}(x, y)\right]$$

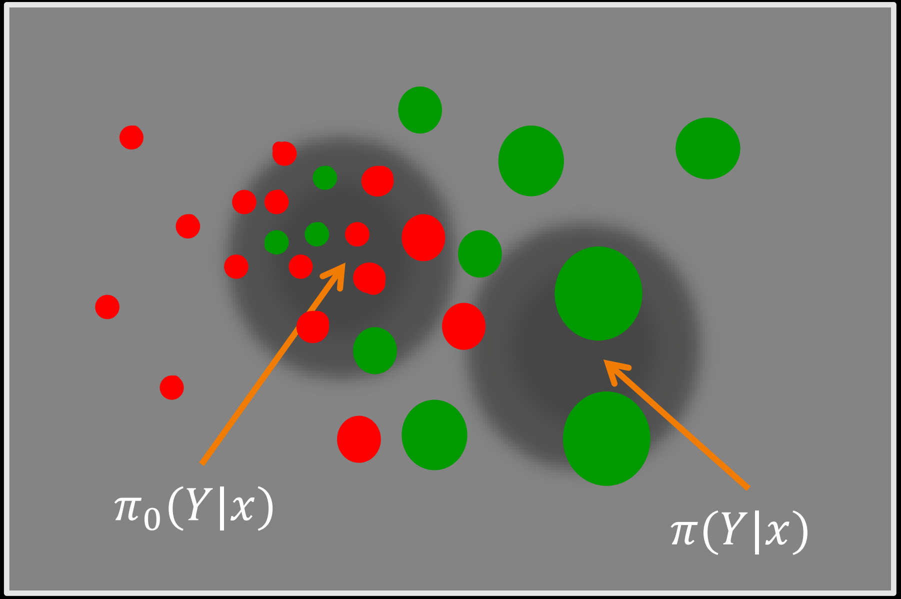


Approach 2: Off-Policy Risk Evaluation

Given $S = \left((x_1, y_1, \delta_1), \dots, (x_n, y_n, \delta_n)\right)$ collected under π_0,

$$\hat{R}(\pi) = \frac{1}{n} \sum_{i=1}^{n} \delta_i \frac{\pi(y_i|x_i)}{\pi_0(y_i|x_i)}$$

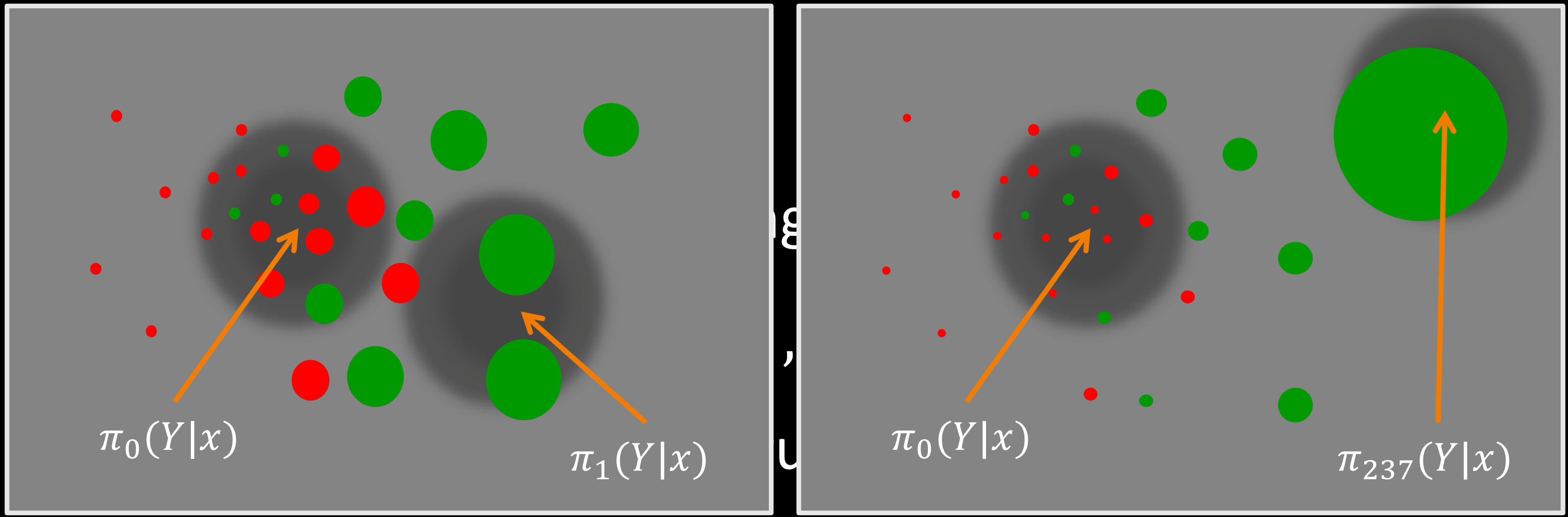
Propensity p_i



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

[Horvitz & Thompson, 1952] [Rubin, 1983] [Zadrozny et al., 2003] [Langford, Li, 2009.]

Partial Information Empirical Risk Minimization



- Training

$$\hat{\pi} := \operatorname{argmin}_{\pi \in H} \sum_{i}^{n} \frac{\pi(y_i|x_i)}{p_i} \, \delta_i$$

[Zadrozny et al., 2003] [Langford & Li], [Bottou, et al., 2014]

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 $1 - \eta$

$$\mathrm{R}(\pi) \leq \hat{R}(\pi) + O\left(\sqrt{\widehat{Var}(\pi)/n}\right) + O(C)$$

Unbiased Estimator

Variance Control

Capacity Control

$$\hat{R}(\pi) = \widehat{Mean}\left(\frac{\pi(y_i|x_i)}{p_i}\delta_i\right)$$

$$\widehat{Var}(\pi) = \widehat{Var}\left(\frac{\pi(y_i|x_i)}{p_i}\delta_i\right)$$

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

[Swaminathan & Joachims, 2015]

Counterfactual Risk Minimization

- Theorem [Generalization Error Bound]

$$R(\pi) \leq \hat{R}(\pi) + O\left(\sqrt{\widehat{Var}(\pi)/n}\right) + O(C)$$

→ Constructive principle for designing learning algorithms

$$\boxed{\pi^{crm} = \underset{\pi \in H_i}{\operatorname{argmin}}\, \hat{R}(\pi) + \lambda_1 \left(\sqrt{\widehat{Var}(\pi)/n}\right) + \lambda_2 C(H_i)}$$

$$\hat{R}(\pi) = \frac{1}{n}\sum_{i}^{n} \frac{\pi(y_i|x_i)}{p_i}\delta_i \qquad \widehat{Var}(\pi) = \frac{1}{n}\sum_{i}^{n}\left(\frac{\pi(y_i|x_i)}{p_i}\,\delta_i\right)^2 - \hat{R}(\pi)^2$$

[Swaminathan & Joachims, 2015]

Outline of Lecture

- Batch Learning from Bandit Feedback (BLBF)

$$S = \left((x_1, y_1, \delta_1), \ldots, (x_n, y_n, \delta_n)\right)$$

→ Find new policy π that selects y with better δ

- Learning Principle for BLBF
  - Hypothesis Space, Risk, Empirical Risk, and Overfitting
  - Learning Principle: Counterfactual Risk Minimization
- Learning Algorithms for BLBF
  - POEM: Bandit training of CRF policies for structured outputs
  - BanditNet: Bandit training of deep network policies

POEM Hypothesis Space

Hypothesis Space: Stochastic policies

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

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: $S = \big((x_1, y_1, \delta_1, p_1), \ldots, (x_n, y_n, \delta_n, p_n)\big)$
  - Hypothesis space: $\pi_w(y|x) = \exp\big(w \cdot \phi(x, y)\big)/Z(x)$
  - Training objective: Let $z_i(w) = \pi_w(y_i|x_i)\delta_i/p_i$

$$w = \underset{w \in \Re^N}{\operatorname{argmin}} \left[\frac{1}{n}\sum_{i=1}^{n} z_i(w) + \lambda_1 \sqrt{\left(\frac{1}{n}\sum_{i=1}^{n} z_i(w)^2\right) - \left(\frac{1}{n}\sum_{i=1}^{n} z_i(w)\right)^2} + \lambda_2 ||w||^2 \right]$$

Unbiased Risk Estimator

Variance Control

Capacity Control

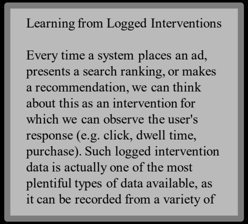
[Swaminathan & Joachims, 2015]

POEM Experiment
Multi-Label Text Classification

- Data: $S = \left((x_1, y_1, \delta_1, p_1), \ldots, (x_n, y_n, \delta_n, p_n)\right)$

  Reuters LYRL RCV1 (top 4 categories)

x_i

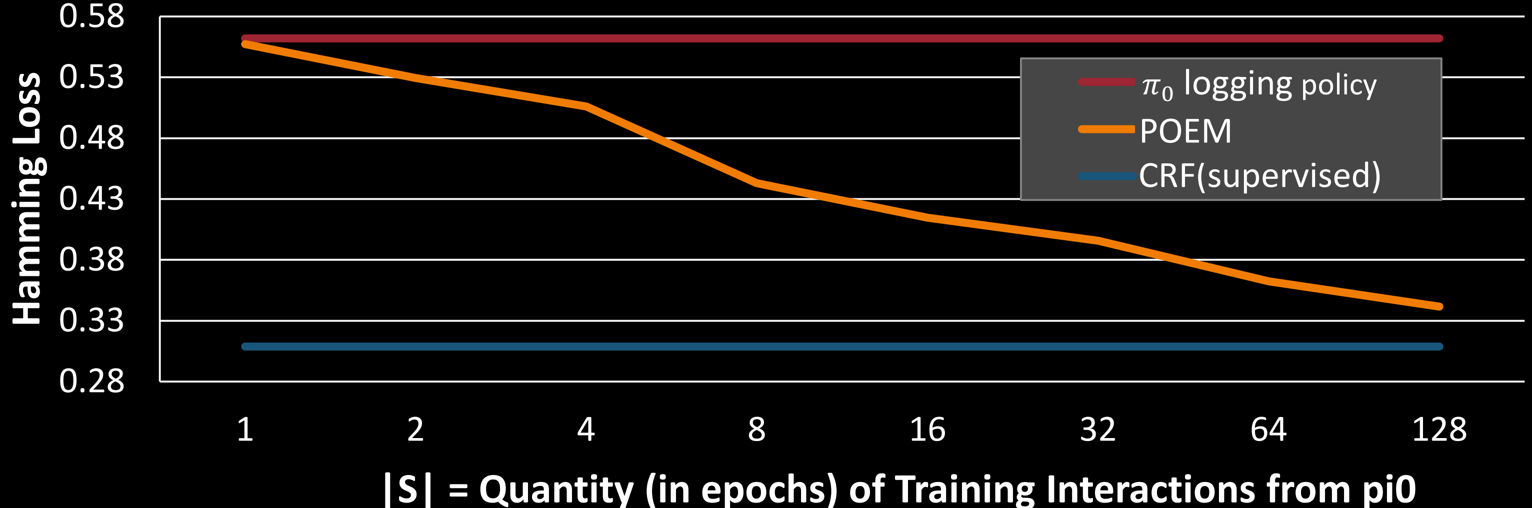


Learning from Logged Interventions

Every time a system places an ad, presents a search ranking, or makes a recommendation, we can think about this as an intervention for which we can observe the user's response (e.g. click, dwell time, purchase). Such logged intervention data is actually one of the most plentiful types of data available, as it can be recorded from a variety of

$\xrightarrow{\pi_0}$ $y_i = \{\text{sport, politics}\}$ $p_i = 0.3$ $\longrightarrow$ $\delta_i = 1$

- Results: POEM with H isomorphic to CRF with one weight vector per label



[Swaminathan & Joachims, 2015]

Does Variance Regularization Improve Generalization?

- IPS: $w = \underset{w \in \Re^N}{\operatorname{argmin}} \left[\hat{R}(w) + \lambda_2 ||w||^2 \right]$
- POEM: $w = \underset{w \in \Re^N}{\operatorname{argmin}} \left[\hat{R}(w) + \lambda_1 \left(\sqrt{\widehat{Var}(w)/n} \right) + \lambda_2 ||w||^2 \right]$

Hamming Loss	Scene	Yeast	TMC	LYRL
π_0	1.543	5.547	3.445	1.463
IPS	1.519	4.614	3.023	1.118
POEM	**1.143**	**4.517**	**2.522**	**0.996**
# examples	4*1211	4*1500	4*21519	4*23149
# features	294	103	30438	47236
# labels	6	14	22	4

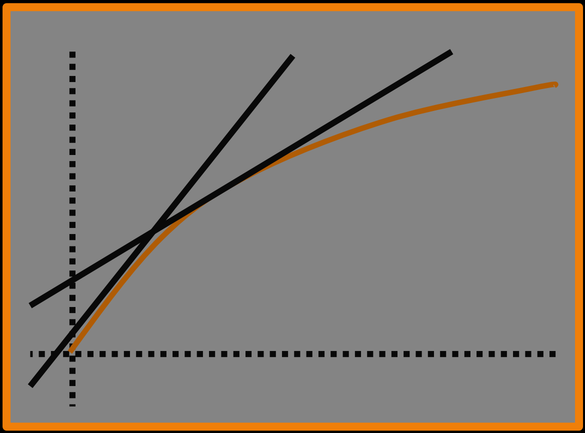
POEM Efficient Training Algorithm

- Training Objective:

$$OPT = \min_{w \in \Re^N} \left[\frac{1}{n} \sum_{i=1}^{n} z_i(w) + \lambda_1 \sqrt{\left(\frac{1}{n} \sum_{i=1}^{n} z_i(w)^2 \right) - \left(\frac{1}{n} \sum_{i=1}^{n} z_i(w) \right)^2} \right]$$

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

$$OPT \leq \min_{w \in \Re^N} \left[\frac{1}{n} \sum_{i=1}^{n} A_i \, z_i(w) + B_i \, z_i(w)^2 \right]$$



- Algorithm:
  - Majorize objective at current w_t
  - Solve majorizing objective via Adagrad to get w_{t+1}

[De Leeuw, 1977+] [Groenen et al., 2008] [Swaminathan & Joachims, 2015]

Counterfactual Risk Minimization

- Theorem [Generalization Error Bound]

$$\mathrm{R}(\pi) \leq \hat{R}(\pi) + O\left(\sqrt{\widehat{Var}(\pi)/n}\right) + O(C)$$

→ Constructive principle for designing learning algorithms

$$\boxed{\pi^{crm} = \underset{\pi \in H_i}{\operatorname{argmin}}\, \hat{R}(\pi) + \lambda_1 \left(\sqrt{\widehat{Var}(\pi)/n}\right) + \lambda_2 C(H_i)}$$

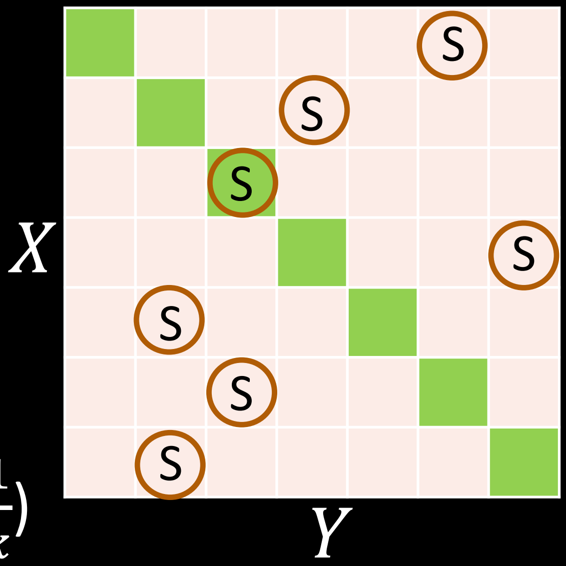
$$\hat{R}(\pi) = \frac{1}{n}\sum_{i}^{n} \frac{\pi(y_i|x_i)}{p_i}\delta_i \qquad \widehat{Var}(\pi) = \frac{1}{n}\sum_{i}^{n}\left(\frac{\pi(y_i|x_i)}{p_i}\,\delta_i\right)^2 - \hat{R}(\pi)^2$$

[Swaminathan & Joachims, 2015]

Propensity Overfitting Problem

- Example
  - Instance Space $X = \{1, \dots, k\}$
  - Label Space $Y = \{1, \dots, k\}$
  - Loss $\delta(x, y) = \begin{cases} -2 & if\ y == x \\ -1 & otherwise \end{cases}$
  - Training data: uniform x,y sample ($p_i = \frac{1}{k}$)
  - Hypothesis space: all deterministic functions

    → $\pi_{opt}(x) = x$ with risk $R(\pi_{opt}) = -2$



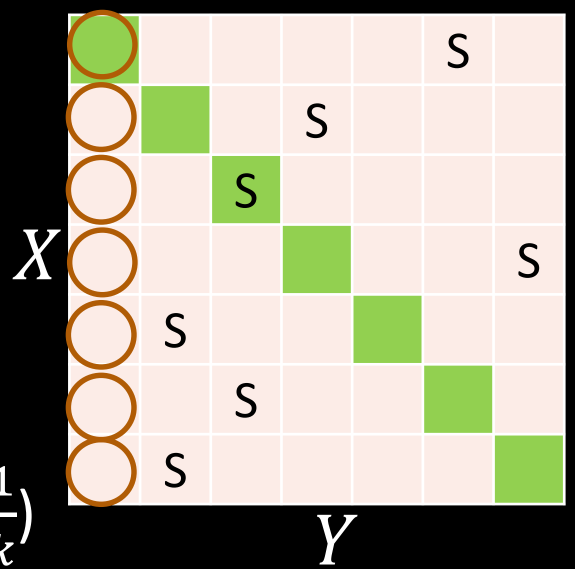
$$R(\hat{\pi}) = \min_{\pi \in H} \frac{1}{n} \sum_{i}^{n} \frac{\pi(y_i | x_i)}{p_i} \delta_i =$$

→ Problem 1: Unbounded risk estimate!

Propensity Overfitting Problem

- Example
  - Instance Space $X = \{1, \dots, k\}$
  - Label Space $Y = \{1, \dots, k\}$
  - Loss $\delta(x, y) = \begin{cases} \cancel{-2}\ 0 \quad if\ y == x \\ \cancel{-1}\ 1 \quad otherwise \end{cases}$
  - Training data: uniform x,y sample ($p_i = \frac{1}{k}$)
  - Hypothesis space: all deterministic functions

    → $\pi_{opt}(x) = x$ with risk $R(\pi_{opt}) = \cancel{-2}\ 0$



$$R(\hat{\pi}) = \min_{\pi \in H} \frac{1}{n} \sum_{i}^{n} \frac{\pi(y_i | x_i)}{p_i} \delta_i =$$

→ Problem 2: Lack of equivariance!

Control Variate

- Idea: Inform estimate when expectation of correlated random variable is known.
  - Estimator:
  $$\hat{R}(\pi) = \frac{1}{n}\sum_{i}^{n}\frac{\pi(y_i|x_i)}{p_i}\delta_i$$
  - Correlated RV with known expectation:
  $$\hat{S}(\pi) = \frac{1}{n}\sum_{i}^{n}\frac{\pi(y_i|x_i)}{p_i}$$

$$E\left[\hat{S}(\pi)\right] = \frac{1}{n}\sum_{i}^{n}\int\frac{\pi(y_i|x_i)}{\pi_0(y_i|x_i)}\pi_0(y_i|x_i)P(x)dy_i dx_i = 1$$

→ Alternative Risk Estimator: Self-normalized estimator

$$\hat{R}^{SN}(\pi) = \frac{\hat{R}(\pi)}{\hat{S}(\pi)}$$

[Hesterberg, 1995] [Swaminathan & Joachims, 2015]

SNIPS Learning Objective

- Method:
  - Data: $S = \big((x_1, y_1, \delta_1, p_1), \dots, (x_n, y_n, \delta_n, p_n)\big)$
  - Hypothesis space: $\pi_w(y|x) = \exp\big(w \cdot \phi(x, y)\big)/Z(x)$
  - Training objective:

$$w = \underset{w \in \Re^N}{\operatorname{argmin}} \left[\hat{R}^{SNIPS}(w) + \lambda_1 \sqrt{\widehat{Var}\big(\hat{R}^{SNIPS}(w)\big)} + \lambda_2 ||w||^2 \right]$$

Self-Normalized Risk Estimator

Variance Control

Capacity Control

[Swaminathan & Joachims, 2015]

How well does NormPOEM generalize?

Hamming Loss	Scene	Yeast	TMC	LYRL
π_0	1.511	5.577	3.442	1.459
POEM (IPS)	1.200	4.520	2.152	0.914
POEM (SNIPS)	**1.045**	**3.876**	**2.072**	**0.799**
# examples	4*1211	4*1500	4*21519	4*23149
# features	294	103	30438	47236
# labels	6	14	22	4

Outline of Lecture

- Batch Learning from Bandit Feedback (BLBF)

$$S = \big((x_1, y_1, \delta_1), \ldots, (x_n, y_n, \delta_n)\big)$$

→ Find new policy π that selects y with better δ

- Learning Principle for BLBF
  - Hypothesis Space, Risk, Empirical Risk, and Overfitting
  - Learning Principle: Counterfactual Risk Minimization
- Learning Algorithms for BLBF
  - POEM: Bandit training of CRF policies for structured outputs
  - BanditNet: Bandit training of deep network policies

BanditNet: Hypothesis Space

Hypothesis Space: Stochastic policies

$$\pi_w(y|x) = \frac{1}{Z(x)} \exp(DeepNet(x, y|w))$$

with

- w: parameter tensors to be learned
- Z(x): partition function

Note: same form as Deep Net with softmax output

[Joachims et al., 2017]

BanditNet: Learning Method

- Method:
  - Data: $S = \big((x_1, y_1, \delta_1, p_1), \dots, (x_n, y_n, \delta_n, p_n)\big)$
  - Hypotheses: $\pi_w(y|x) = \exp\big(DeepNet(x|w)\big)/Z(x)$
  - Training objective:

$$w = \underset{w \in \Re^N}{\operatorname{argmin}} \left[\underbrace{\hat{R}^{SNIPS}(w)}_{\text{Self-Normalized Risk Estimator}} + \lambda_1 \underbrace{\sqrt{\widehat{Var}\big(\hat{R}^{SNIPS}(w)\big)}}_{\text{Variance Control}} + \underbrace{\lambda_2 \big||w|\big|^2}_{\text{Capacity Control}} \right]$$

Self-Normalized Risk Estimator

Variance Control

Capacity Control

[Joachims et al., 2017]

BanditNet: Learning Method

- Method:
  - Data: $S = \big((x_1, y_1, \delta_1, p_1), \dots, (x_n, y_n, \delta_n, p_n)\big)$
  - Representation: Deep Network Policies

$$\pi_w(y|x) = \frac{1}{Z(x,w)} \exp\big(DeepNet(y|x,w)\big)$$

  - SNIPS Training Objective:

$$w = \underset{w \in \Re^N}{\operatorname{argmin}} \left[\hat{R}_{SNIPS}(w) + \lambda \big||w|\big|^2\right]$$
$$= \underset{w \in \Re^N}{\operatorname{argmin}} \left[\frac{1}{\sum_{i=1}^{n} \frac{\pi_w(y_i|x_i)}{p_i}} \sum_{i=1}^{n} \frac{\pi_w(y_i|x_i)}{p_i} \delta_i + \lambda \big||w|\big|^2\right]$$

Optimization via SGD

- Problem: SNIPS objective not suitable for SGD
- Step 1: Discretize over values in denominator

$$\hat{w} = \underset{S_j}{\operatorname{argmin}} \left[\underset{\mathrm{w}}{\operatorname{argmin}} \left[\frac{1}{S_j} \sum_{i=1}^{n} \frac{\pi_w(y_i|x_i)}{p_i} \delta_i \right] \text{ subject to } \frac{1}{n} \sum_{i=1}^{n} \frac{\pi_w(y_i|x_i)}{p_i} = S_j \right]$$

- Step 2: View as series of constrained OP

$$\hat{w}_j = \underset{\mathrm{w}}{\operatorname{argmin}} \left[\sum_{i=1}^{n} \frac{\pi_w(y_i|x_i)}{p_i} \delta_i \right] \text{ subject to } \frac{1}{n} \sum_{i=1}^{n} \frac{\pi_w(y_i|x_i)}{p_i} = S_j$$

- Step 3: Eliminate constraint via Lagrangian

$$\hat{w}_j = \underset{\mathrm{w}}{\operatorname{argmin}} \max_{\lambda} \left[\sum_{i=1}^{n} \frac{\pi_w(y_i|x_i)}{p_i} (\delta_i - \lambda) + \lambda S_j \right]$$

Optimization via SGD

- Step 4: Search grid over λ instead of S_j
  - Hard: Given S_j, find λ_j.
  - Easy: Given λ_j, find S_j.

→Solve $$\hat{w}_j = \underset{\mathrm{w}}{\operatorname{argmin}} \left[\sum_{i=1}^{n} \frac{\pi_w(y_i|x_i)}{p_i} (\delta_i - \lambda_j) + \lambda_j S_j \right]$$

→Compute $$S_j = \frac{1}{n} \sum_{i=1}^{n} \frac{\pi_w(y_i|x_i)}{p_i}$$

BanditNet: Training Algorithm

- Given:
  - Data: $S = \left((x_1, y_1, \delta_1, p_1), \dots, (x_n, y_n, \delta_n, p_n)\right)$
  - Lagrange Multipliers: $\lambda_j \in \{\lambda_1, \dots, \lambda_k\}$
- Compute:
  - For each λ_j solve: $\hat{w}_j = \operatorname*{argmin}_{\mathrm{w}} \left[\sum_{i=1}^{n} \frac{\pi_w(y_i|x_i)}{p_i}(\delta_i - \lambda_j)\right]$
  - For each $\hat{\mathrm{w}}_j$ compute: $S_j = \frac{1}{n}\sum_{i=1}^{n} \frac{\pi_{\hat{w}_j}(y_i|x_i)}{p_i}$
  - Find overall $\hat{w}$: $\hat{w} = \operatorname*{argmin}_{\hat{w}_j, S_j} \left[\frac{1}{S_j}\sum_{i=1}^{n} \frac{\pi_{\hat{w}_j}(y_i|x_i)}{p_i}\delta_i\right]$

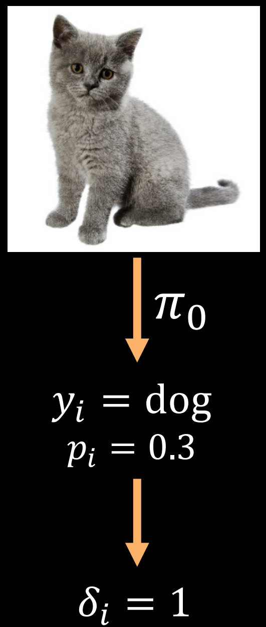
Object Recognition: Data and Setup

- Data: CIFAR-10 (fully labeled)

  → $S^* = \big((x_1, y_1^*), \ldots, (x_m, y_m^*)\big)$

- Bandit feedback generation:
  - Draw image x_i
  - Use logging policy $\pi_0(Y|x_i)$ to predict y_i
    - Record propensity $\pi_0(Y = y_i|x_i)$
  - Observe loss $\delta_i = [y_i \neq y_i^*]$

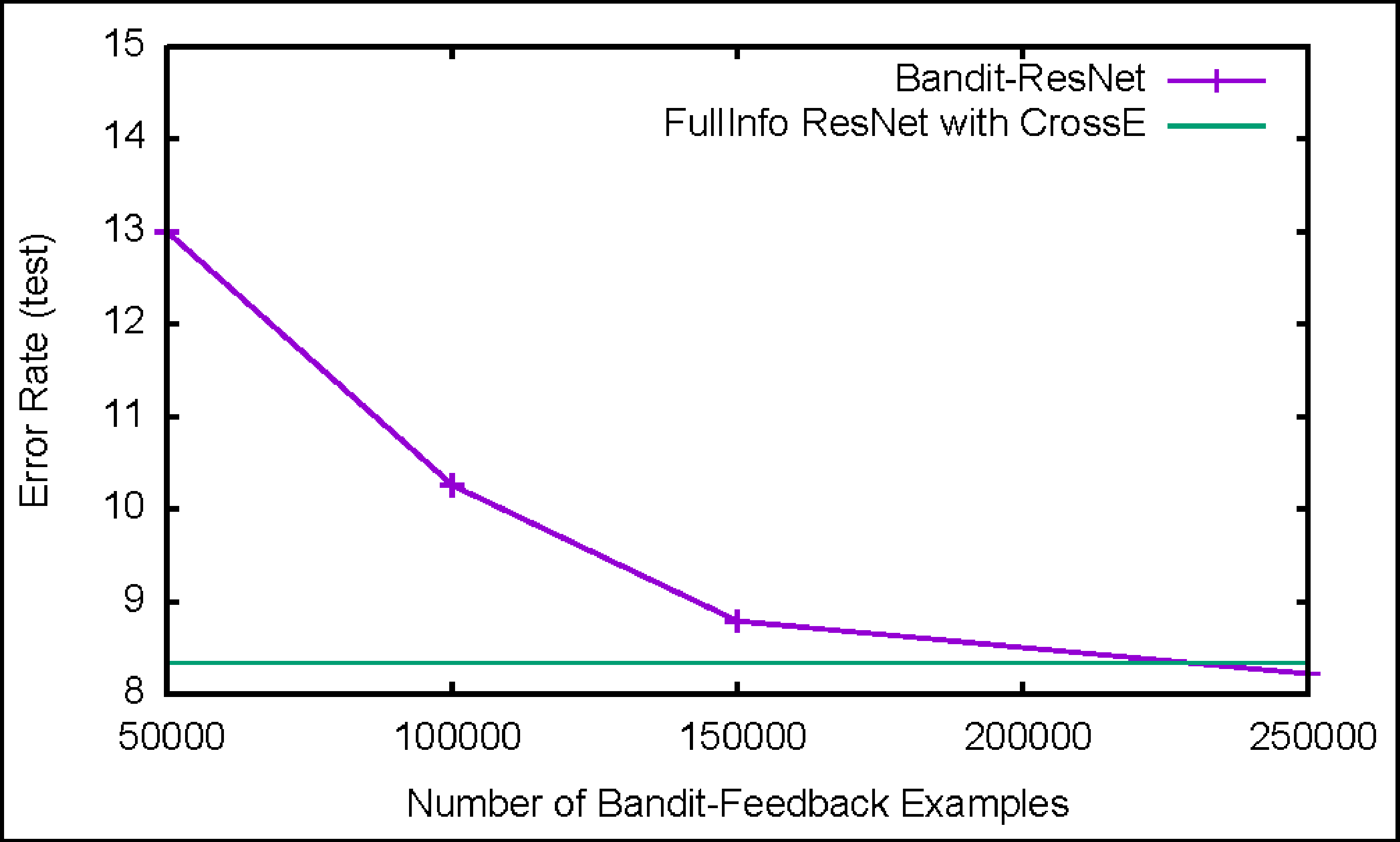
  → $S = \big((x_1, y_1, \delta_1, p_1), \ldots, (x_n, y_n, \delta_n, p_n)\big)$



- Network architecture: ResNet20 [He et al., 2016]

[Beygelzimer & Langford, 2009] [Joachims et al., 2017]

Bandit Feedback vs. Test Error

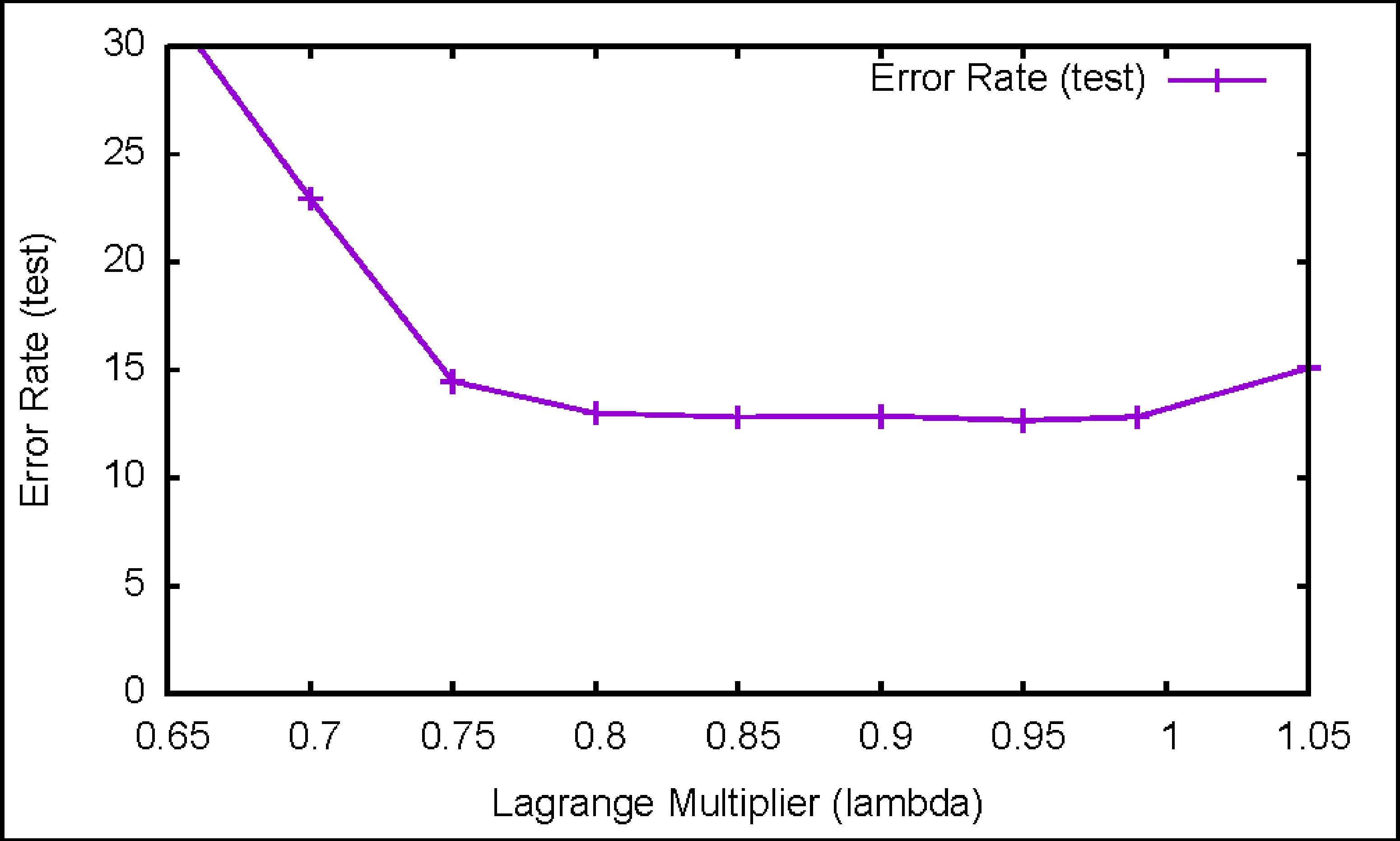


Logging Policy π_0: 49% error rate

Bandit-ResNet with naïve IPS: >49% error rate

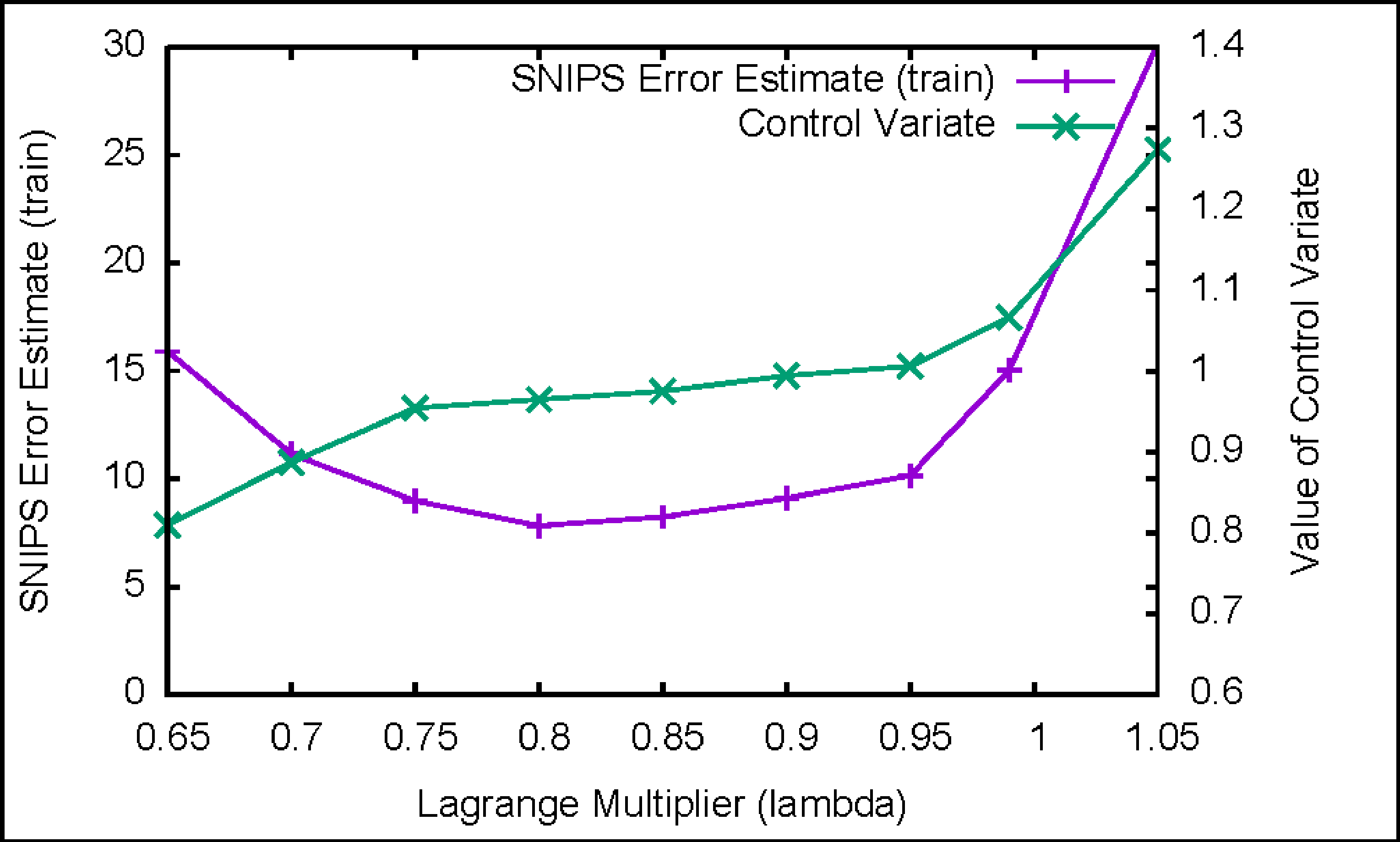
[Joachims et al., 2017]

Lagrange Multiplier vs. Test Error



Large basin of optimality far away from naïve IPS.

Analysis of SNIPS Estimate



Control variate responds to the Lagrange multiplier monotonically.
SNIPS training error resembles test error.

Conclusions and Future

- Batch Learning from Bandit Feedback
  - Feedback for only presented action
    $$S = \big((x_1, y_1, \delta_1, p_1), \dots, (x_n, y_n, \delta_n, p_n)\big)$$
  - Goal: Find new system π that selects y with better δ
  - Learning Principle for BLBF: Counterfactual Risk Minimization
- Learning from Logged Interventions: BLBF and Beyond
  - POEM: [Swaminathan & Joachims, 2015c]
  - NormPOEM: [Swaminathan & Joachims, 2015c]
  - BanditNet: [Joachims et al., 2018]
  - SVM PropRank [Joachims et al., 2017a]
  - DeepPropDCG: [Agarwal et al., 2018]
  - Unbiased Matrix Factorization: [Schnabel et al. 2016]
- Future Research
  - Other learning algorithms? Other partial-information settings?
  - How to handle new bias-variance trade-off in risk estimators?
  - Applications
- Software, Papers, SIGIR Tutorial, Data: www.joachims.org