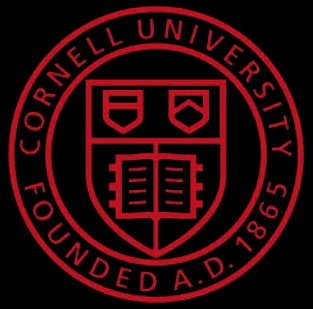
Learning from Logged Interventions

Adith Swaminathan, Damien Lefortier, Maarten De Rijke,
Artem Grotov, Xiaotao Gu, Thorsten Joachims

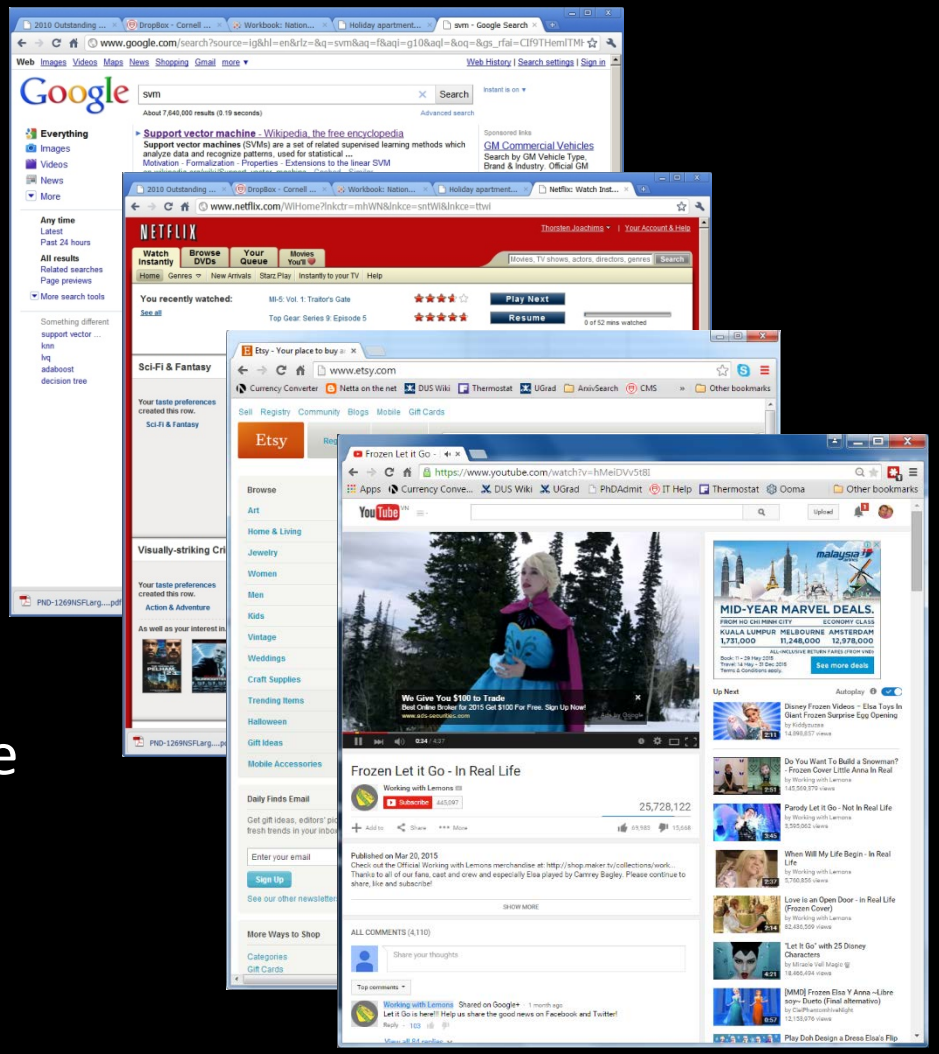
Department of Computer Science
Department of Information Science
Cornell University



Funded in part through NSF Awards IIS-1247637, IIS-1513692, IIS-1615706.

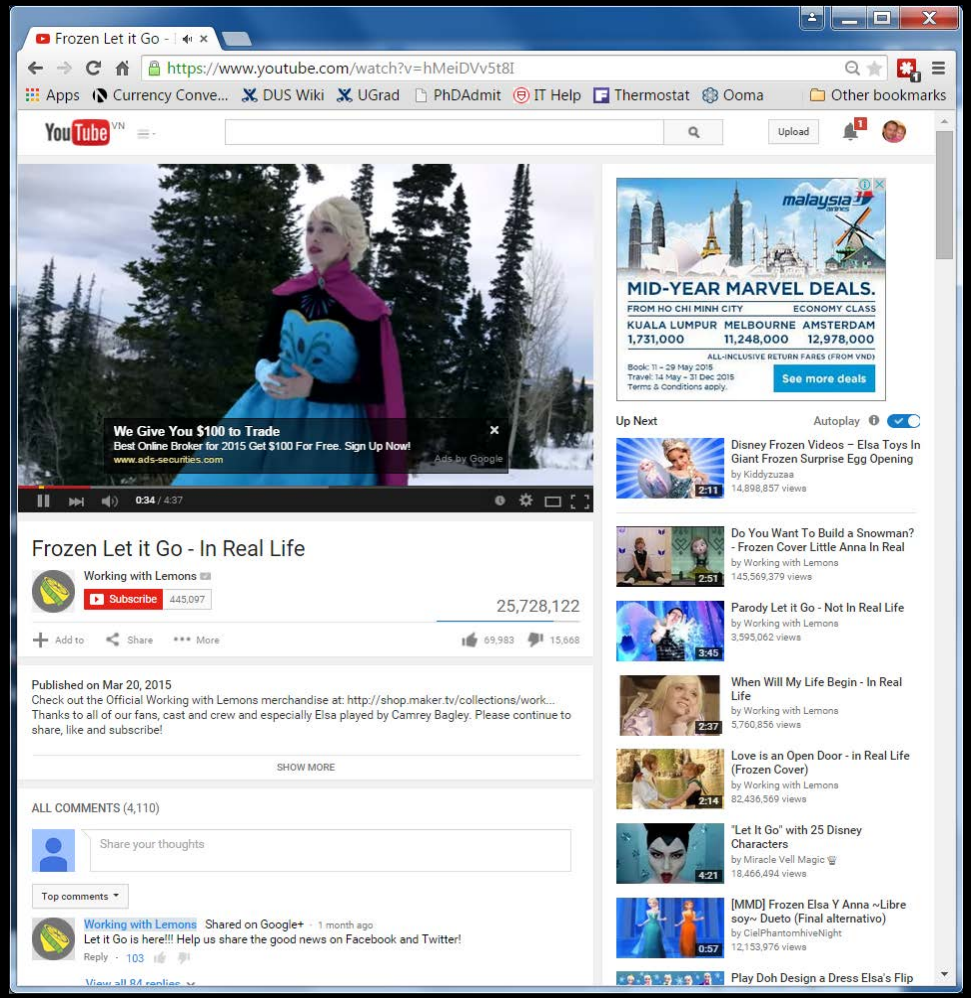
Interactive Systems

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



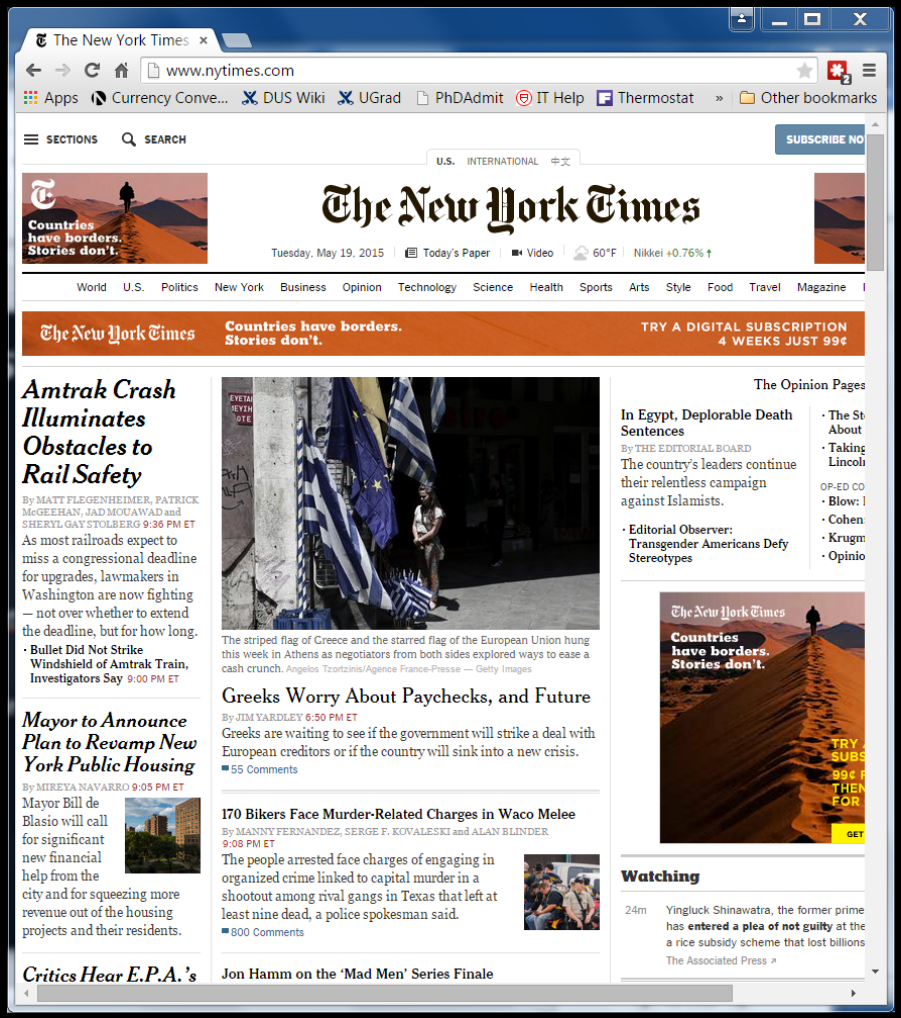
Historic Interaction Logs: Ad Placement

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



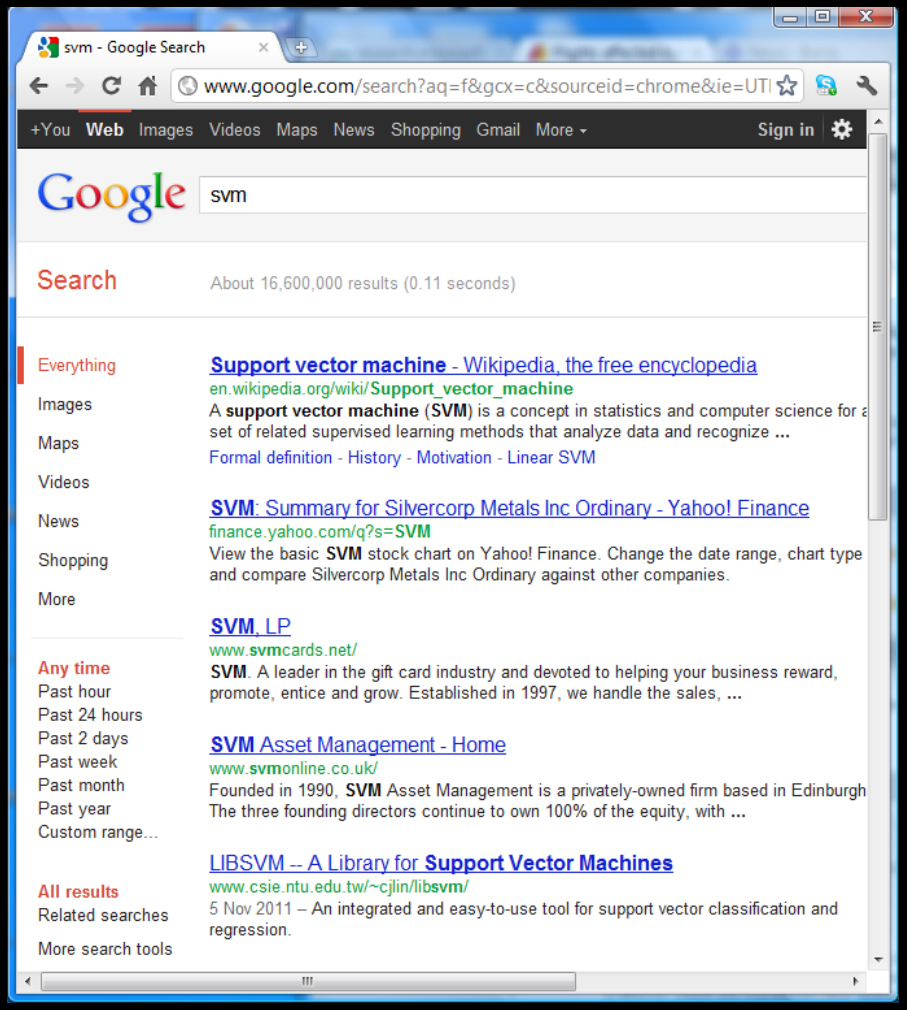
Historic Interaction Logs: News Recommender

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



Historic Interaction Logs: Search Engine

- Context x:
  - Query
- Action y:
  - Ranking
- Feedback $\delta(x, y)$:
  - Clicks on SERP



Batch Learning from Bandit Feedback

- Data

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

→ "Bandit" Feedback

- Properties
  - Contexts x_i drawn i.i.d. from unknown $P(X)$
  - Actions y_i selected by existing system π_0: $X \rightarrow Y$
  - Feedback 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.	?

Outline of Talk

- Batch Learning from Bandit Feedback (BLBF)

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

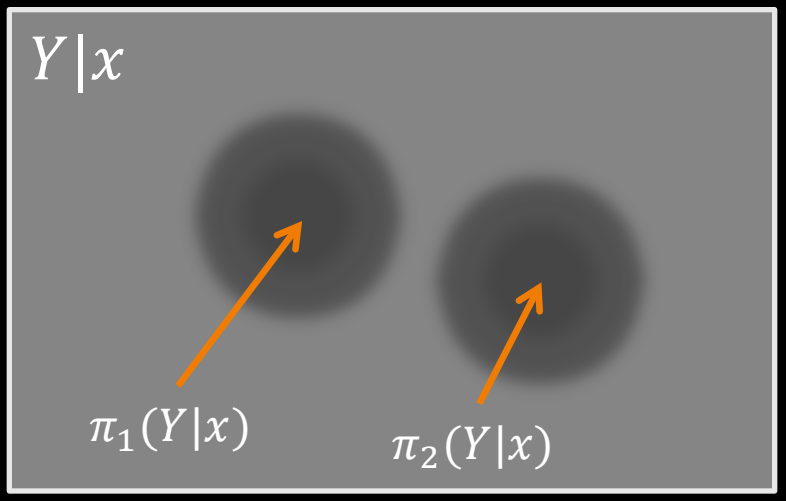
  → Find new policy π that selects y with better δ

- 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 for Structured Output Prediction
  - BanditNet: Bandit training of Deep Networks
- Application: Display Advertising

Hypothesis Space

Definition [Stochastic Hypothesis / Policy]:

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



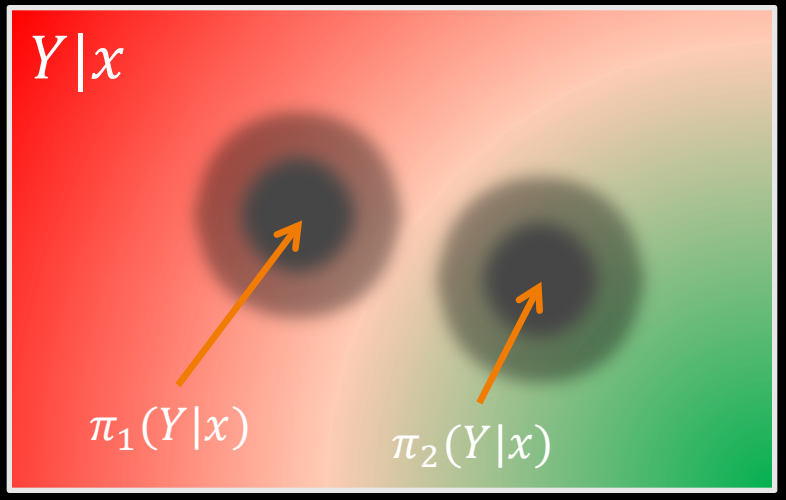
Note: stochastic prediction rules $\supset$ deterministic prediction rules

Risk

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

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

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



On-Policy Risk Estimation

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

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

→ A/B Testing

Field π_1: Draw $x \sim P(x)$, predict $y \sim \pi_1(Y|x\ )$, get $\delta(x, y)$

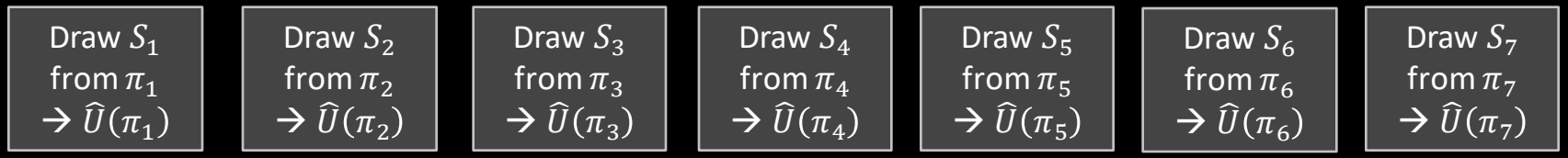
Field π_2: Draw $x \sim P(x)$, predict $y \sim \pi_2(Y|x\ )$, get $\delta(x, y)$

⋮

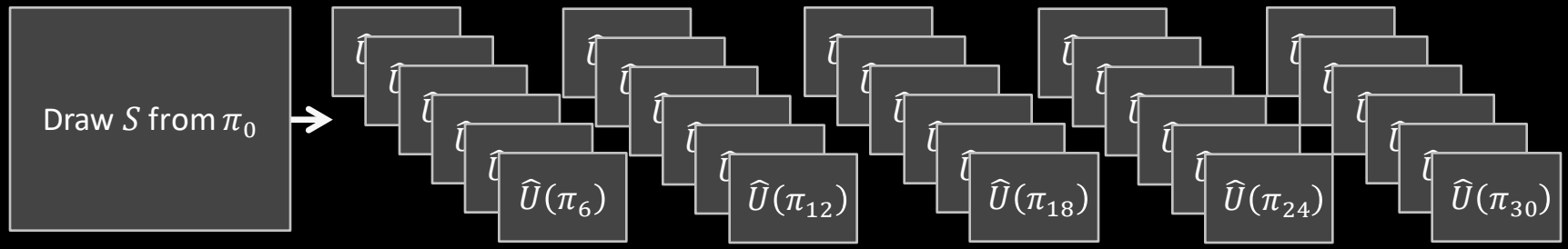
Field $\pi_{|H|}$: Draw $x \sim P(x)$, predict $y \sim \pi_{|H|}(Y|x)$, get $\delta(x, y)$

Evaluating Online Metrics Offline

- Online: On-policy A/B Test

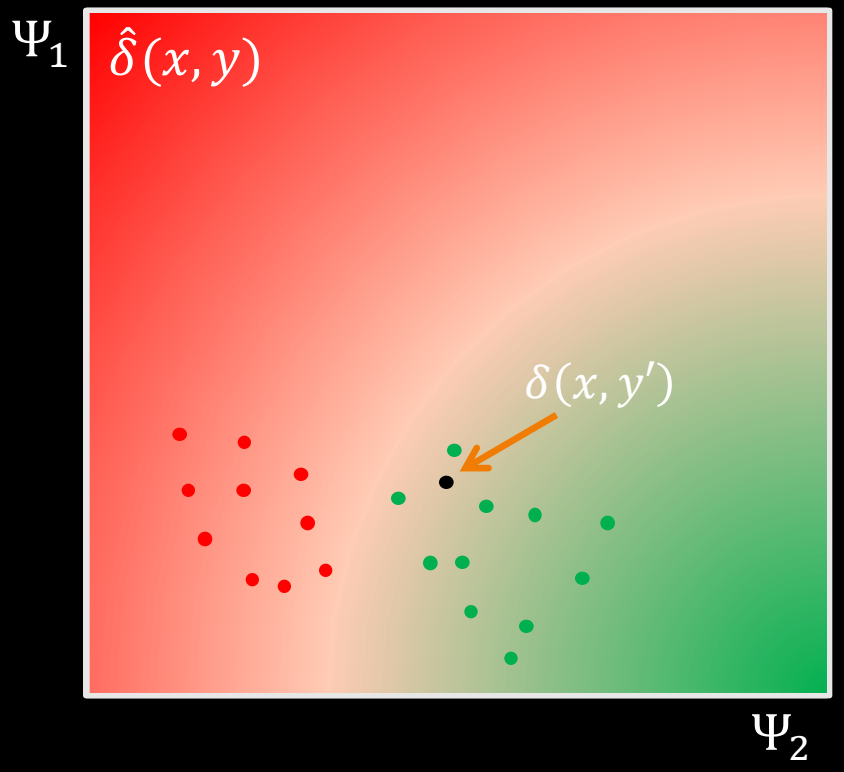


- Offline: Off-policy Counterfactual Estimates



Approach 1: Reward Predictor

- Data:
  $S = \left((x_1, y_1, \delta_1), \ldots, (x_n, y_n, \delta_n)\right)$
- Idea:
  - Use data from π_0 to learn reward predictor $\hat{\delta}(x, y)$
- Learn $\hat{\delta}: x \times y \to \Re$
  1. Represent via features $\Psi(x, y)$
  2. Learn regression based on $\Psi(x, y)$ from S collected under π_0
  3. Predict $\hat{\delta}(x, y')$ for $y' = \pi(x)$ of new policy π

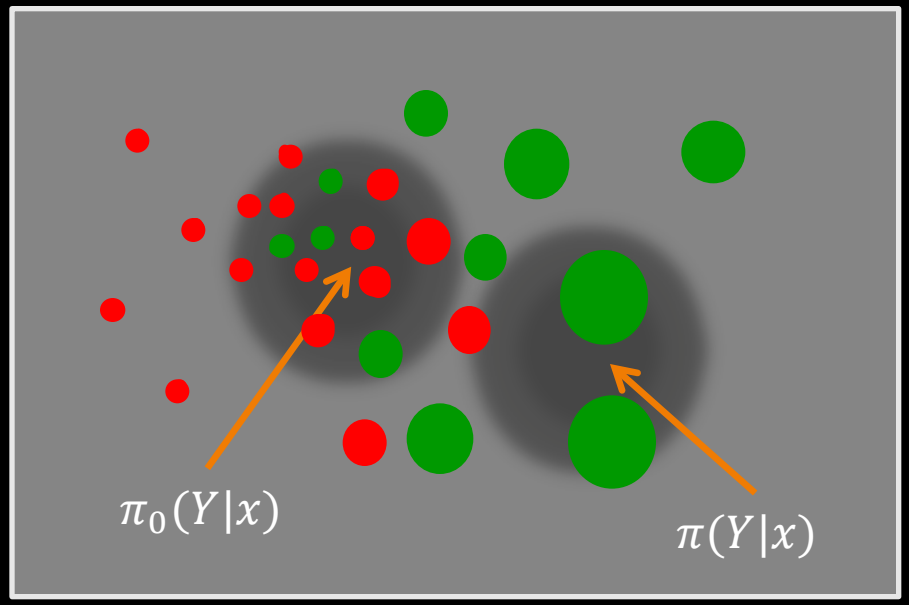


Approach 2: Off-Policy Risk Evaluation

Given $S = \big((x_1, y_1, \delta_1), \ldots, (x_n, y_n, \delta_n)\big)$ collected under π_0,

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

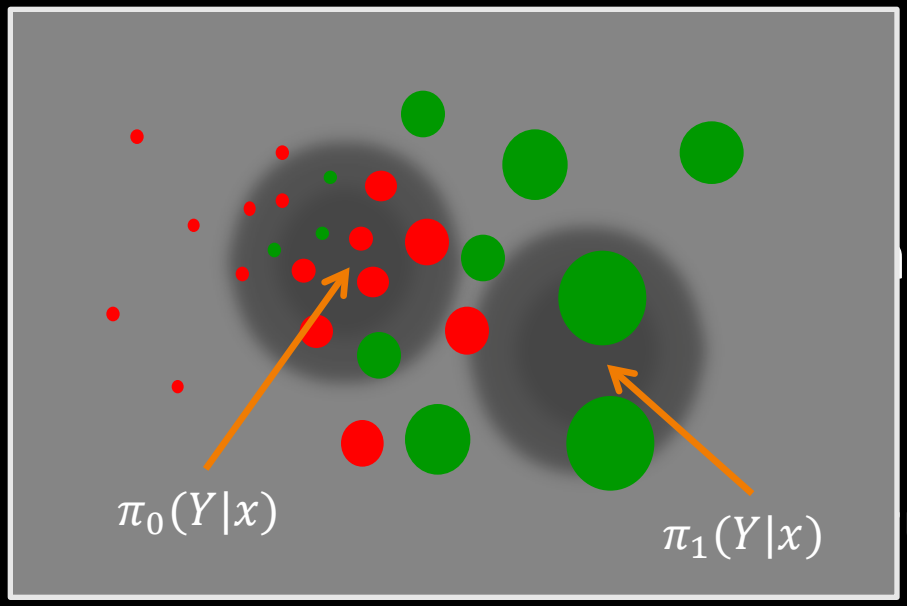
Propensity p_i

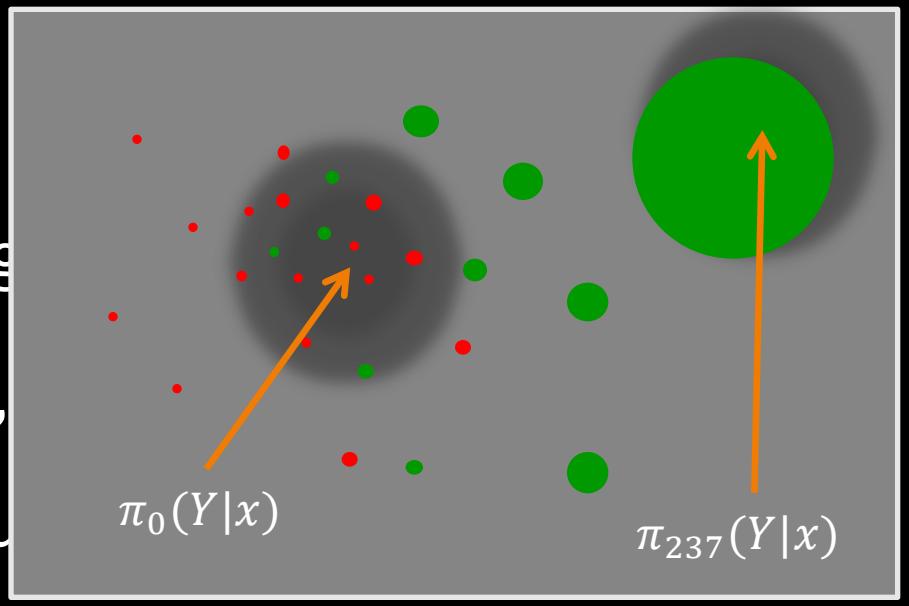


→ Unbiased estimate of risk, if propensity nonzero everywhere (where it matters).

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

Partial Information Empirical Risk Minimization





- Training

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

[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]

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

→ Constructive principle for designing learning algorithms

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

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

[Swaminathan & Joachims, 2015]

Outline of Talk

- 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 for Structured Output Prediction
  - BanditNet: Bandit training of Deep Networks
- Application: Display Advertising

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: $S = \left((x_1, y_1, \delta_1, p_1), \ldots, (x_n, y_n, \delta_n, p_n)\right)$
  - Hypothesis space: $\pi_w(y|x) = \exp\left(w \cdot \phi(x, y)\right)/Z(x)$
  - 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

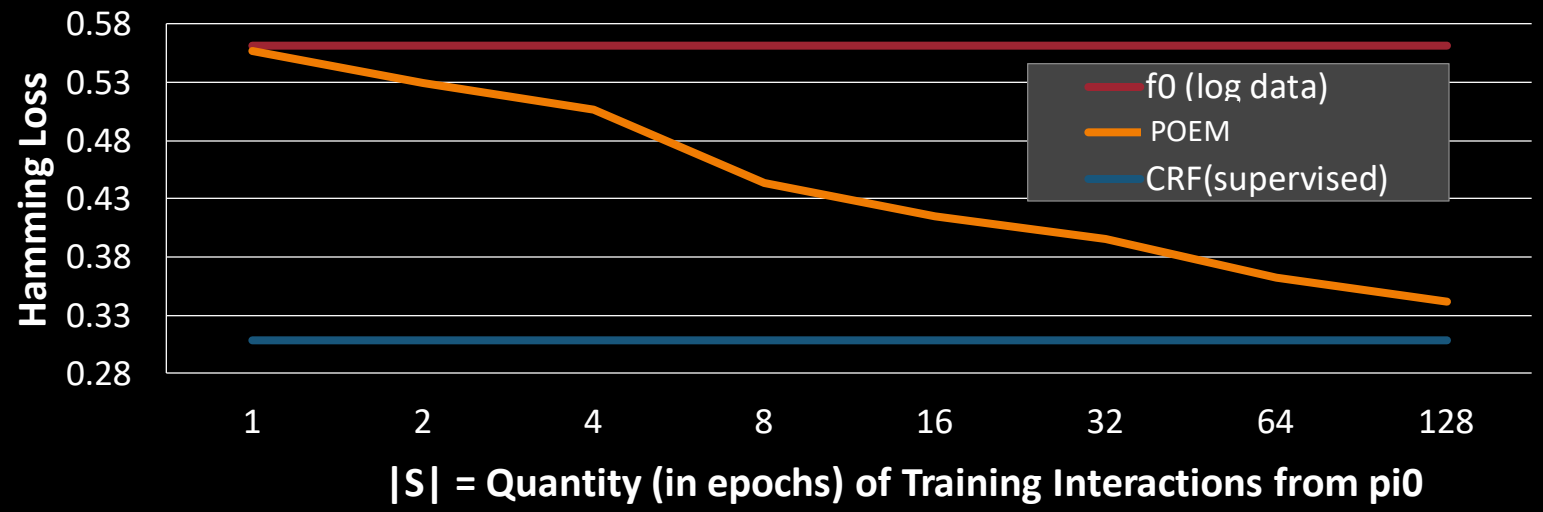
Variance Control

Capacity Control

[Swaminathan & Joachims, 2015]

POEM Experiment
Multi-Label Text Classification

- Data: $S = \left((x_1, y_1, \delta_1, p_1), \ldots, (x_n, y_n, \delta_n, p_n)\right)$
  - x: Text document
  - y: Predicted label vector
  - δ: number of incorrect labels in y
  - p_n: propensity under logging policy π_0
- Results: Reuters LYRL RCV1 (top 4 categories)
  - POEM with H isomorphic to CRF with one weight vector per label



[Swaminathan & Joachims, 2015]

Does Variance Regularization Improve Generalization?

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

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

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

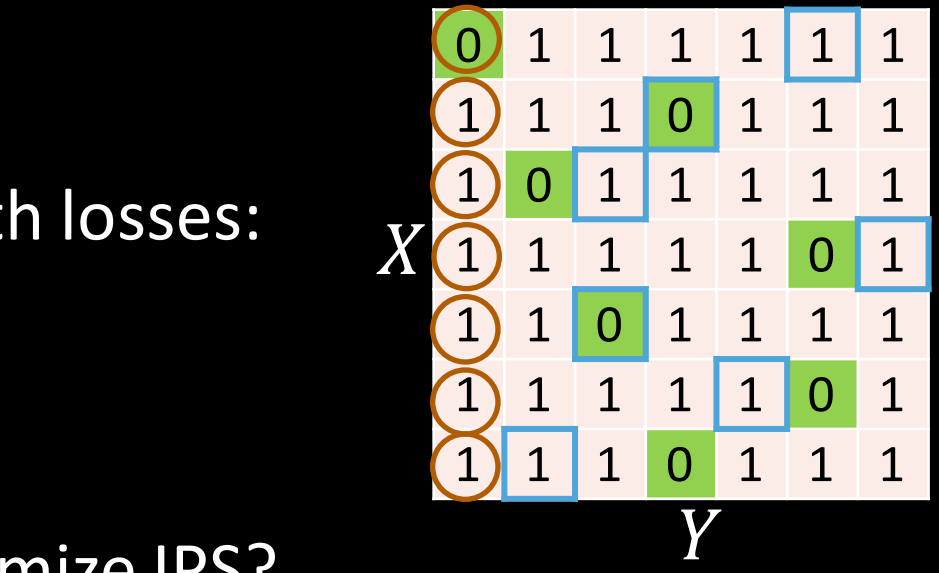
$$\pi^{crm} = \underset{\pi \in H_i}{\operatorname{argmin}}\, \hat{R}(\pi) + \lambda_1 \left(\sqrt{\widehat{Var}(\pi)/n}\right) + \lambda_2 C(H_i)$$

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

[Swaminathan & Joachims, 2015]

Problem: Propensity Overfitting

- Example
  - Training sample with losses:



  - Which $\pi(y|x)$ minimize IPS?

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

→ Avoid the training observations!

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

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

NormPOEM Learning Method

- Method:
  - 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 $z_i(w) = \pi_w(y_i|x_i)\delta_i/p_i$

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

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	1.200	4.520	2.152	0.914
NormPOEM	**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

Ad Placement: Data and Setup

- Criteo Ad-Placement
  - Task:
    - For user x, pick product y from candidate set
    - Size of candidate set: ~10 products
  - Performance measure δ:
    - click through rate
  - Data:
    - 21M examples from stochastic production system with logged propensities
    - Feature vector $\phi(x, y)$: about 70k binary features
  - Experiment Setup:
    - Train/Validation/Test split
    - Pick hyperparameters on Validation Set via IPS estimator
    - Use IPS estimator to estimate true click through rate using test set
  - Dataset publicly available [Lefortier et al., 2016]

Ad Placement: Results

- Criteo Ad-Placement
  - Task: pick product from candidate set

Test-Set Click Rate ($* 10^4$) Est	
Random	44.7 ± 2.1
π_0	53.5 ± 0.2
Click-Predictor	48.4 ± 3.2
IPS	54.1 ± 2.5
DoublyRobust	57.4 ± 14.0
NormPOEM	**58.0 ± 3.4**

  - Performance measure: click through rate
  - Data: stochastic logging with production system

Outline of Talk

- 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 for Structured Output Prediction
  - BanditNet: Bandit training of Deep Networks
- Application: Display Advertising

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

BanditNet: Learning Method

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

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

Self-Normalized Risk Estimator

Variance Control

Capacity Control

Ad Placement: BanditNet Results

- Criteo Ad-Placement
  - Task: pick product from candidate set

Test-Set Click Rate ($* 10^4$) Est	
Random	44.7 ± 2.1
π_0	53.5 ± 0.2
Click-Predictor	48.4 ± 3.2
IPS	54.1 ± 2.5
DoublyRobust	57.4 ± 14.0
NormPOEM	58.0 ± 3.4
BanditNet	**58.8**

  - BanditNet: 2-Layer, RELU, 100 Hidden Units
  - Pick NumEpochs and LagrangeMultiplier on validation set, no other parameter tuning yet

Conclusions and Future

- Batch Learning from Bandit Feedback
  - Feedback for only presented action

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

  - Goal: Find new system π that selects y with better δ
- Learning Principle for BLBF
  - Counterfactual Risk Minimization
  - Self-Normalized Risk Estimator
- Learning Methods for BLBF
  - POEM: [Swaminathan & Joachims, 2015c]
  - NormPOEM: [Swaminathan & Joachims, 2015c]
  - BanditNet: [Swaminathan, Grotov, DeRijke, Joachims, forthcoming]
- Future Research
  - Other learning algorithms?
  - Other risk estimators? See poster tonight [Agarwal & Joachims, 2017]
  - How to handle new bias-variance trade-off in risk estimators?
- Software, Papers, SIGIR 2016 Tutorial, Criteo Data: www.joachims.org