Counterfactual Model for Learning 2

CS6780 – Advanced Machine Learning

Spring 2019

Thorsten Joachims

Cornell University

Reading:

G. Imbens, D. Rubin, Causal Inference for Statistics …, 2015. Chapters 1,3,12.

From Evaluation to Learning

Setting: Batch Learning from Bandit Feedback (BLBF)

- Naïve "Model the World" Learning:
  - Learn: $\hat{\delta}: x \times y \rightarrow \Re$
  - Derive Policy:

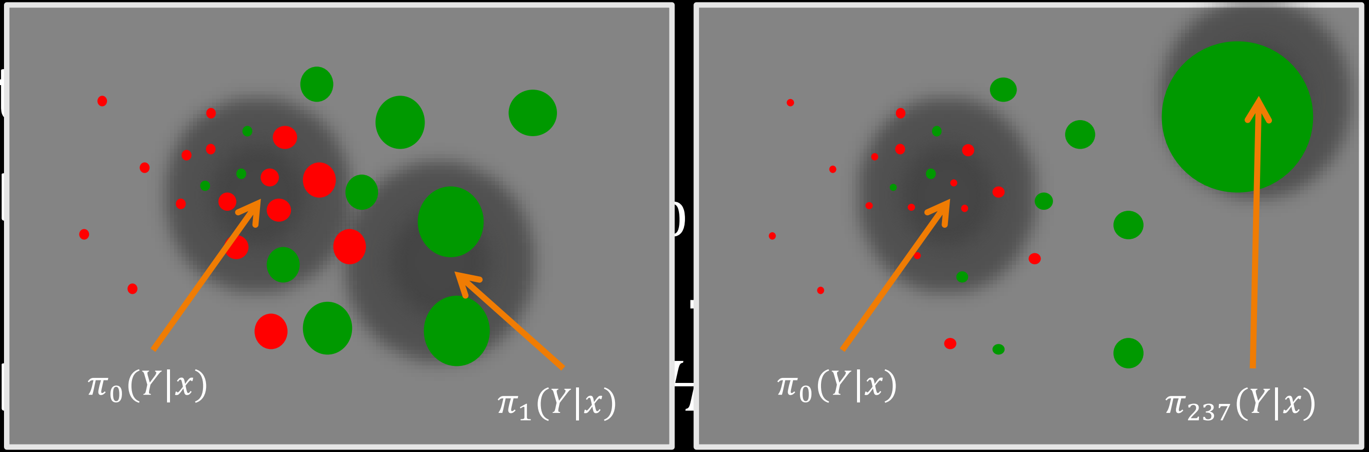
$$\pi(y|x) = \underset{y'}{\operatorname{argmin}}\left[\hat{\delta}(x, y')\right]$$

- Naïve "Model the Bias" Learning:
  - Find policy that optimizes IPS training error

$$\pi = \underset{\pi'}{\operatorname{argmin}}\left[\sum_i \frac{\pi'(y_i|x_i)}{\pi_0(y_i|x_i)} \delta_i\right]$$

Partial-Information ERM

- Set
  –
  –



- Training

$$\hat{\pi} := \text{argmax}_{\pi \in H} \left[\sum_{i=1}^{n} \frac{\pi(y_i|x_i)}{\pi_0(y_i|x_i)} \delta_i \right]$$

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

Learning Theory for BLBF

Theorem [Generalization Error Bound]

For any policy space H with capacity C, and for all $\pi \in H$ with probability $1 - \eta$

$$\mathrm{U}(\pi) \geq \hat{U}(\pi) - O\left(\sqrt{\frac{\widehat{Var}\left(\hat{U}(\pi)\right)}{n}}\right) - O(C)$$

Unbiased Estimator

Variance Overfitting

Capacity Overfitting

→ 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

Constructive principle for learning algorithms

→ Maximize learning theoretical bound

$$\pi^{crm} = \underset{\pi \in H}{\operatorname{argmax}} \left[\hat{U}(\pi) - \lambda_1 \left(\sqrt{\widehat{Var}\left(\hat{U}(\pi)\right)/n} \right) - \lambda_2 C(H) \right]$$

Training Error/Utility (IPS)

Variance Regularization

Capacity Regularization

[Swaminathan & Joachims, 2015]

POEM: Policy Space

Policy space

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

with

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

Note: same form as CRF or Structural SVM

POEM: Learning Method

Policy Optimizer for Exponential Models (POEM)

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

$$w = \underset{w \in \Re^N}{\operatorname{argmax}} \left[\hat{U}(\pi_w) - \lambda_1 \left(\sqrt{\widehat{Var}\left(\hat{U}(\pi_w)\right)} \right) - \lambda_2 ||w||^2 \right]$$

IPS Estimator

Variance Regularization

Capacity Regularization

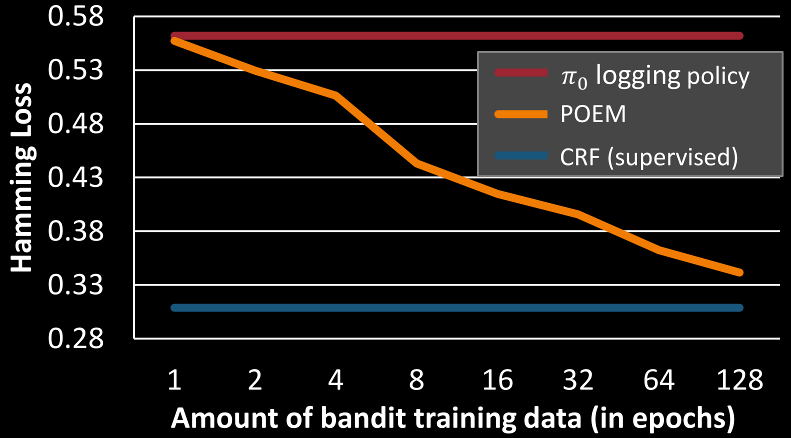
[Swaminathan & Joachims, 2015]

POEM: Text Classification

Data: Reuters Text Classification

- $S^* = ((x_1, y_1^*), \ldots, (x_m, y_m^*))$
- Label vectors $y^* = (y^1, y^2, y^3, y^4)$

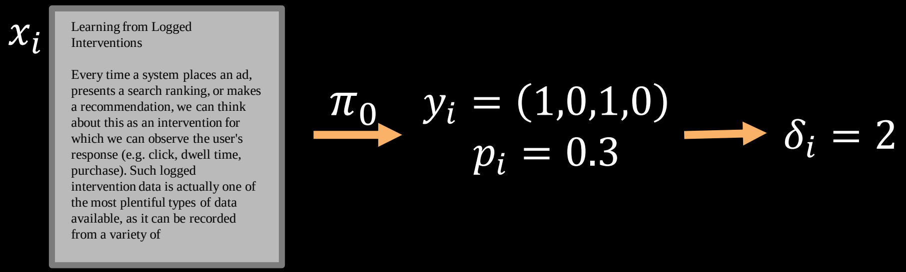
Results:



Bandit feedback generation:

- Draw document x_i
- Pick y_i via logging policy $\pi_0(Y|x_i)$
- Observe loss $\delta_i = \text{Hamming}(y_i, y_i^*)$

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



[Joachims et al., 2017]

BanditNet: Policy Space

Policy space

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

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

- Deep networks with bandit feedback (BanditNet):
  - Data: $S = \big((x_1, y_1, \delta_1, p_1), \ldots, (x_n, y_n, \delta_n, p_n)\big)$
  - Hypotheses: $\pi_w(y|x) = \exp\big(DeepNet(x|w)\big)/Z(x)$

$$w = \underset{w \in \Re^N}{\operatorname{argmax}} \left[\hat{U}(\pi_w) - \lambda_1 \left(\sqrt{\widehat{Var}\left(\hat{U}(\pi_w)\right)} \right) - \lambda_2 ||w||^2 \right]$$

Self-Normalized IPS Estimator

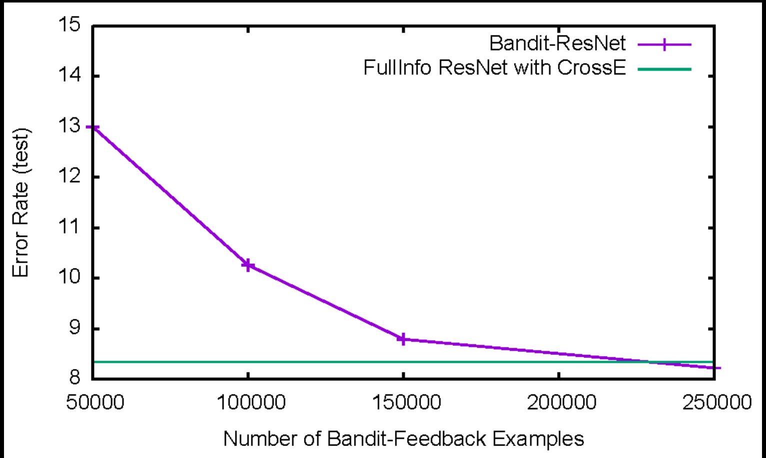
Variance Regularization

Capacity Regularization

[Joachims et al., 2018]

BanditNet: Object Recognition

- Data: CIFAR-10
  - $S^* = ((x_1, y_1^*), \dots, (x_m, y_m^*))$
  - ResNet20 [He et al., 2016]
- Results



- Bandit feedback generation:
  - Draw image x_i
  - Pick y_i via logging policy $\pi_0(Y|x_i)$
  - Observe loss $\delta_i = [y_i \neq y_i^*]$

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

π_0 → $y_i = \text{dog}$, $p_i = 0.3$ → $\delta_i = 1$

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