

# Contextual Bandit Algorithms with Supervised Learning Guarantees

AISTATS 2011

Alina Beygelzimer  
IBM Research

John Langford  
Yahoo! Research

Lihong Li  
Yahoo! Research

Lev Reyzin  
Georgia Tech

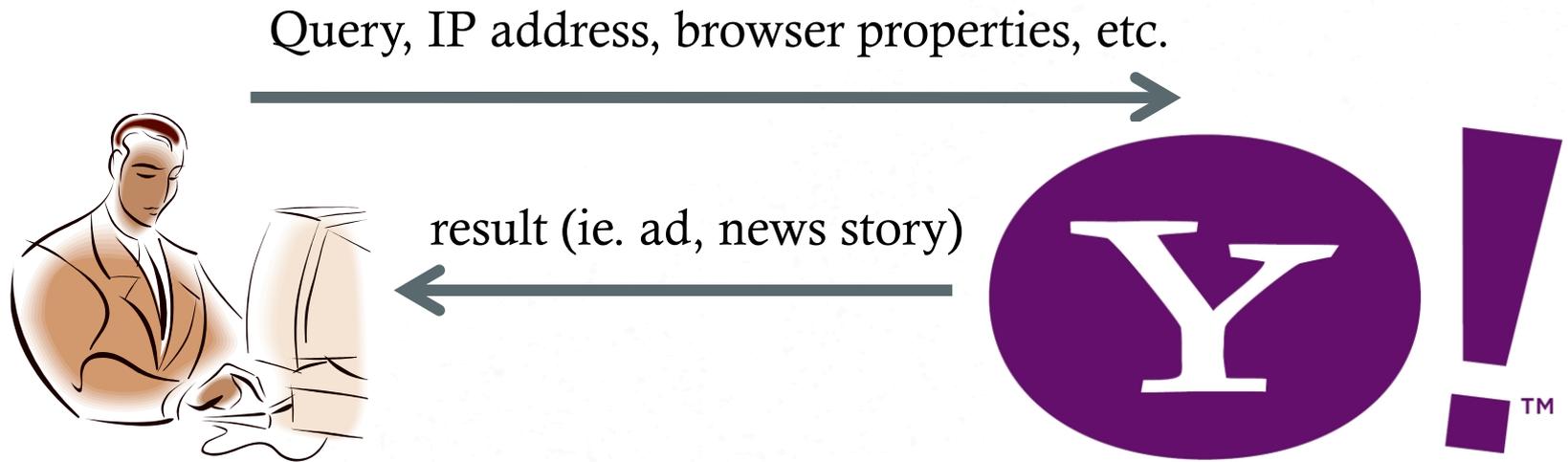
Robert E. Schapire  
Princeton University

# Serving Content to Users

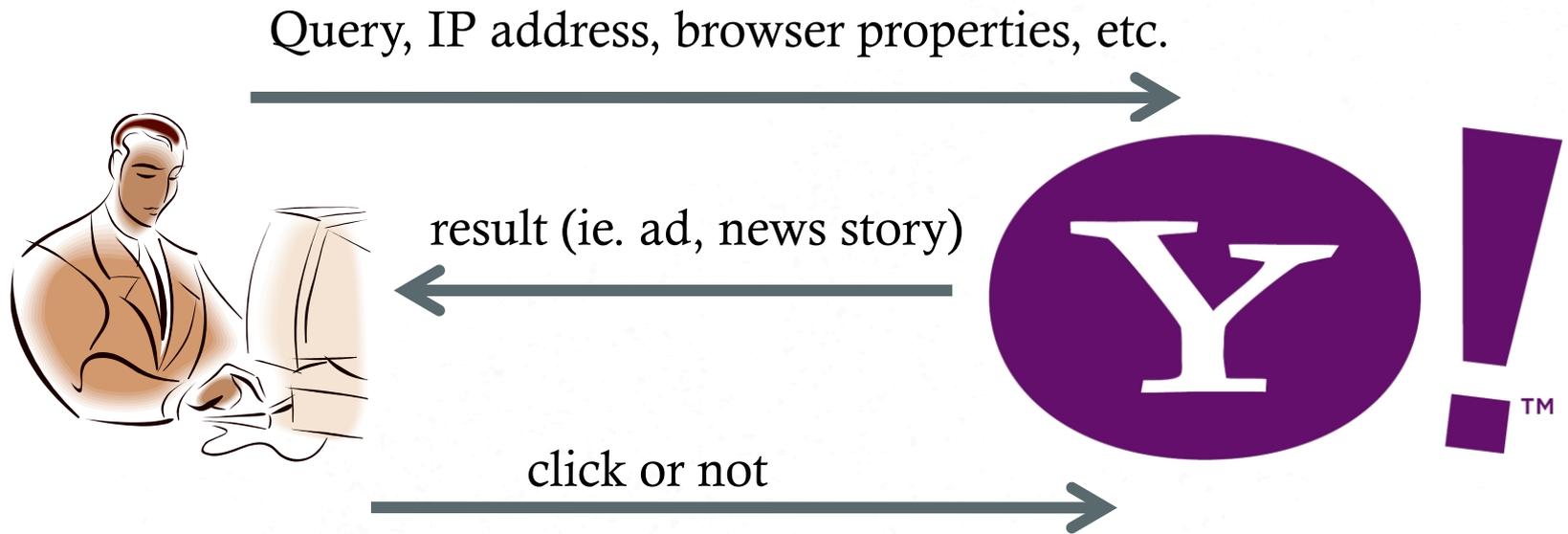
Query, IP address, browser properties, etc.



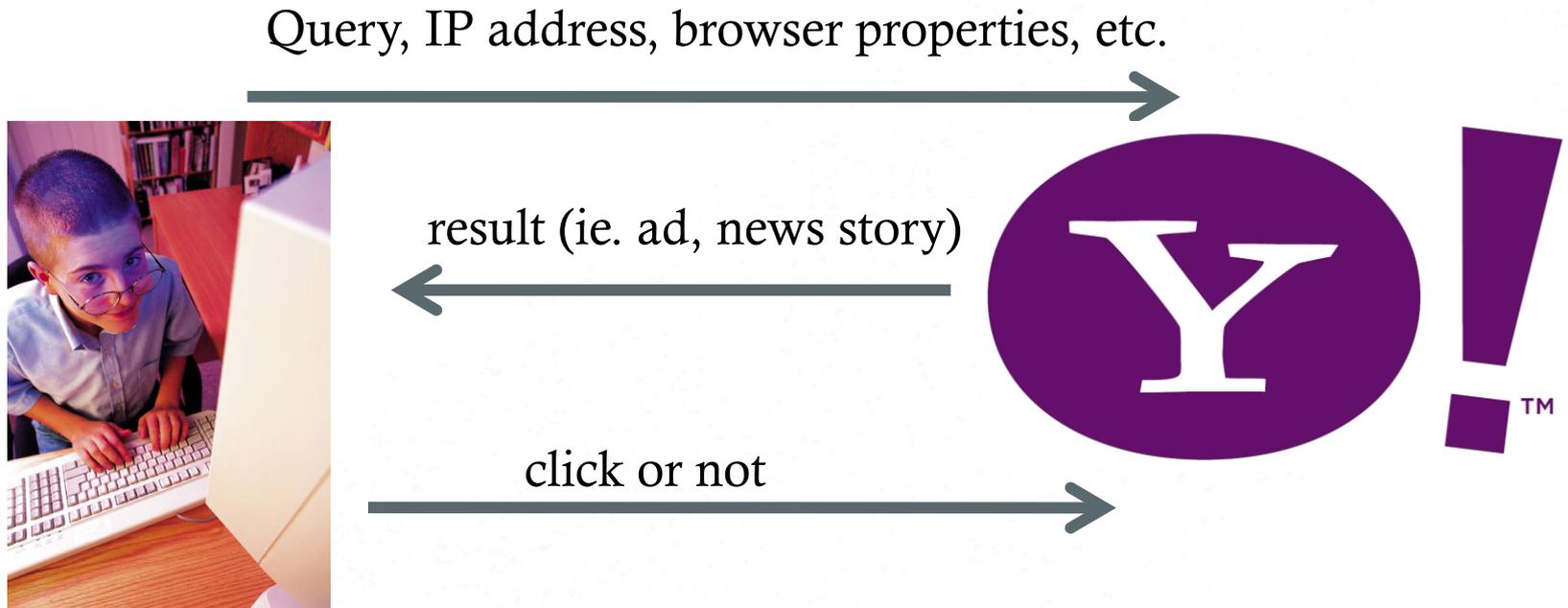
# Serving Content to Users



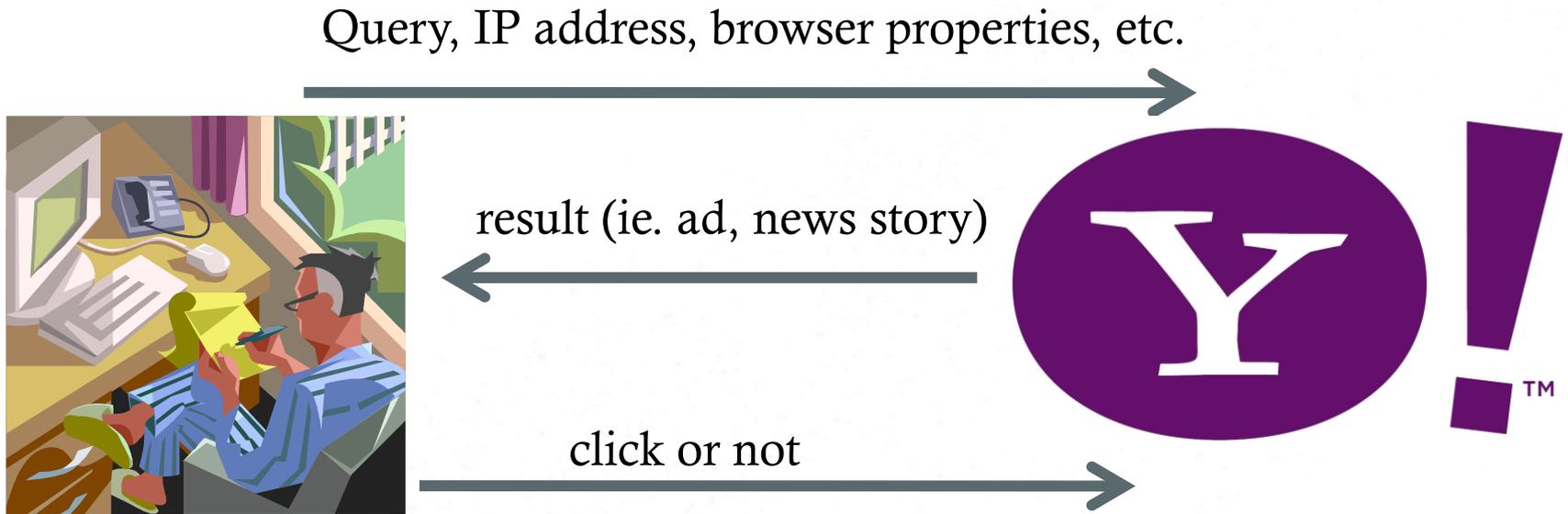
# Serving Content to Users



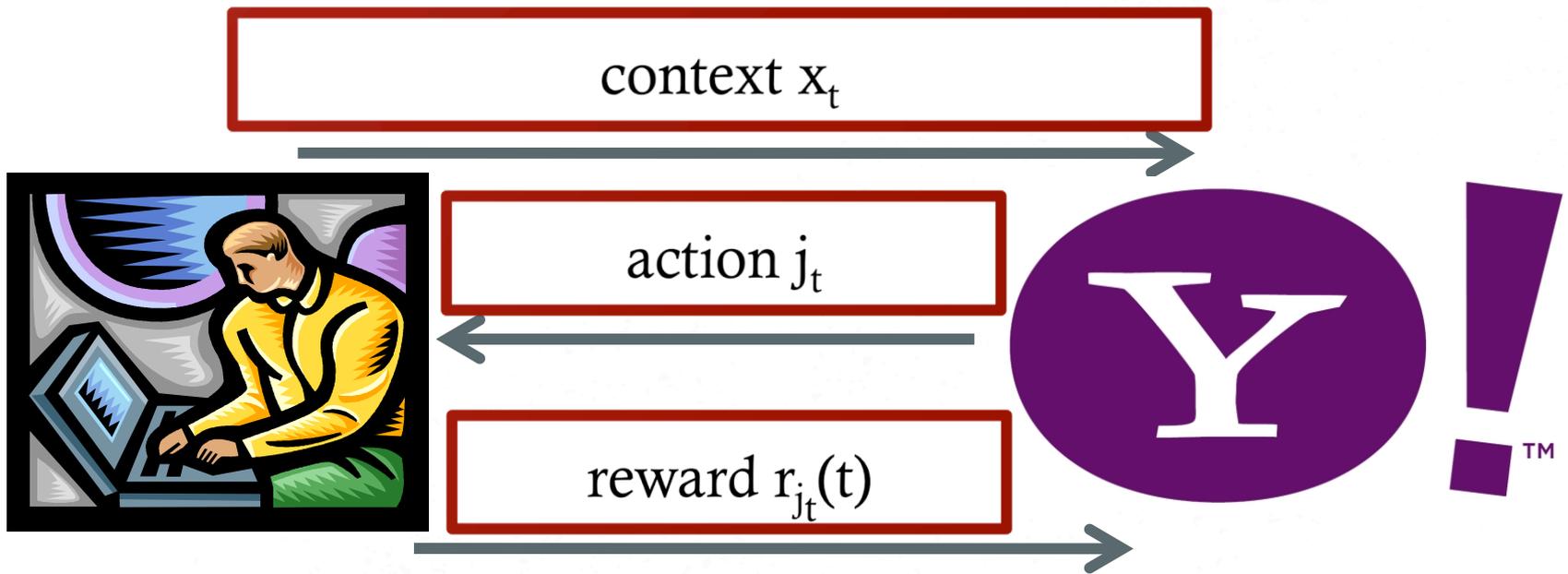
# Serving Content to Users



# Serving Content to Users



# Serving Content to Users



# Outline

---

- **Formally define the setting.**
- Show ideas that fail.
- Give a high probability optimal algorithm.
- Dealing with VC sets.
- Experiments

# The Contextual Bandit Setting

- T rounds, K possible actions, N policies  $\pi$  in  $\Pi$  (context  $\rightarrow$  actions)
- for  $t=1$  to T
  - world commits to rewards  $\mathbf{r}(t)=r_1(t),r_2(t),\dots,r_K(t)$
  - world provides context  $x_t$
  - learner's policies recommend  $\pi_1(x_t), \pi_2(x_t), \dots, \pi_N(x_t)$
  - learner chooses action  $j_t$
  - learner receives reward  $r_{j_t}(t)$
- want to compete with following the best policy in hindsight

# Regret

- **reward** of algorithm A:  $G_A(T) \doteq \sum_{t=1}^T r_{j_t}(t)$
- **expected reward** of policy  $i$ :  $G_i(T) \doteq \sum_{t=1}^T \pi_i(x_t) \cdot r(t)$
- algorithm A's **regret**:  $\max_i G_i(T) - G_A(T)$

# Regret

- algorithm A's regret:  $\max_i G_i(T) - G_A(T)$
- expected regret:  $\max_i G_i(T) - E[G_A(T)]$
- high probability regret:  $P[\max_i G_i(T) - G_A(T) > \varepsilon] \leq \delta$

# Some Observations

- This is harder than supervised learning. In the bandit setting we do not know the rewards of actions not taken.
- This is not the traditional  $K$ -armed bandit setting. In the traditional bandit setting there is no context (or experts).
  - In the simpler  $K$ -armed bandit setting, there is no context. We just compete with best arm in hindsight.
  - The traditional setting is akin to showing everyone the same advertisement, article, etc.

# Previous Results

Algorithm	Regret	High Prob?	Contextual?
Exp4 [ACFS '02]	$\tilde{O}(KT \ln(N))^{1/2}$	No	Yes
$\epsilon$ -greedy, epoch-greedy [LZ '07]	$\tilde{O}((K \ln(N))^{1/3} T^{2/3})$	Yes	Yes
Exp3.P [ACFS '02] UCB [Auer '00]	$\tilde{O}(KT)^{1/2}$	Yes	No

$\Omega(\sqrt{KT})$  lower bound [ACFS '02]

# Our Result

Algorithm	Regret	High Prob?	Contextual?
Exp4 [ACFS '02]	$\tilde{O}(KT \ln(N))^{1/2}$	No	Yes
$\epsilon$ -greedy, epoch-greedy [LZ '07]	$\tilde{O}((K \ln(N)^{1/3})T^{2/3})$	Yes	Yes
Exp3.P [ACFS '02] UCB [Auer '00]	$\tilde{O}(KT)^{1/2}$	Yes	No
Exp4.P [BLLRS '10]	$\tilde{O}(KT \ln(N/\delta))^{1/2}$	Yes	Yes

$\Omega(\sqrt{KT})$  lower bound [ACFS '02]

# Outline

---

- Formally define the setting.
- **Show ideas that fail.**
- Give a high probability optimal algorithm.
- Dealing with VC sets.
- Experiments

# First Some Failed Approaches

- **Bad idea 1:** Maintain a set of plausible hypotheses and randomize uniformly over their predicted actions.

# First Some Failed Approaches

- **Bad idea 1:** Maintain a set of plausible hypotheses and randomize uniformly over their predicted actions.
  - Adversary has two actions, one always paying off 1 and the other 0. Hypothesis generally agree on correct action, except for a different one which defects each round. This incurs regret of  $\sim T/2$ .

# First Some Failed Approaches

- **Bad idea 1:** Maintain a set of plausible hypotheses and randomize uniformly over their predicted actions.
  - Adversary has two actions, one always paying off 1 and the other 0. Hypothesis generally agree on correct action, except for a different one which defects each round. This incurs regret of  $\sim T/2$ .
- **Bad idea 2:** Maintain a set of plausible hypotheses and randomize uniformly among the hypothesis.

# First Some Failed Approaches

- **Bad idea 1:** Maintain a set of plausible hypotheses and randomize uniformly over their predicted actions.
  - Adversary has two actions, one always paying off 1 and the other 0. Hypothesis generally agree on correct action, except for a different one which defects each round. This incurs regret of  $\sim T/2$ .
- **Bad idea 2:** Maintain a set of plausible hypotheses and randomize uniformly among the hypothesis.
  - Adversary has two actions, one always paying off 1 and the other 0. If all but one of  $> 2T$  hypothesis always predict wrong arm, and only 1 hypothesis always predicts good arm, with probability  $> 1/2$  it is never picked and algorithm incurs regret of  $T$ .

# epsilon-greedy

- Rough idea of  $\epsilon$ -greedy (or epoch-greedy [Langford and Zhang '07]): act randomly for  $\epsilon$  rounds, otherwise go with best action (or policy).
- Even if we know the number of rounds in advance, epsilon-first won't get us regret  $O(T)^{1/2}$ , even in the non-contextual setting.
- Rough analysis: even for just 2 arms, we suffer regret:  $\epsilon + (T - \epsilon) / (\epsilon^{1/2})$ .
  - $\epsilon \approx T^{2/3}$  is optimal tradeoff.
  - gives regret  $\approx T^{2/3}$
  - in comparison, in this paper we achieve  $\approx T^{1/2}$

# Outline

---

- Formally define the setting.
- Show ideas that fail.
- **Give a high probability optimal algorithm.**
- Dealing with VC sets.
- Experiments

# Ideas Behind Exp4.P

- **exponential weights**
  - keep a weight on each expert that drops exponentially in the expert's (estimated) performance
- **upper confidence bounds**
  - use an upper confidence bound on each expert's estimated reward
- **ensuring exploration**
  - make sure each action is taken with some minimum probability
- **importance weighting**
  - give rare events more importance to keep estimates unbiased

# Exponential Weight Algorithm for Exploration and Exploitation with Experts

(EXP4) [Auer et al. '95]

(slide from Beygelzimer & Langford ICML 2010 tutorial)

Initialization:  $\forall \pi \in \Pi : w_t(\pi) = 1$

For each  $t = 1, 2, \dots$ :

1. Observe  $x_t$  and let for  $a = 1, \dots, K$

$$p_t(a) = (1 - Kp_{\min}) \frac{\sum_{\pi} \mathbf{1}[\pi(x_t) = a] w_t(\pi)}{\sum_{\pi} w_t(\pi)} + p_{\min},$$

where  $p_{\min} = \sqrt{\frac{\ln |\Pi|}{KT}}$ .

2. Draw  $a_t$  from  $p_t$ , and observe reward  $r_t(a_t)$ .
3. Update for each  $\pi \in \Pi$

$$w_{t+1}(\pi) = \begin{cases} w_t(\pi) \exp\left(p_{\min} \frac{r_t(a_t)}{p_t(a_t)}\right) & \text{if } \pi(x_t) = a_t \\ w_t(\pi) & \text{otherwise} \end{cases}$$

# Exponential Weight Algorithm for Exploration and Exploitation with Experts

(Exp4.P) [Beygelzimer, Langford, Li, R, Schapire '10]

Initialization:  $\forall \pi \in \Pi : w_t(\pi) = 1$

For each  $t = 1, 2, \dots$ :

1. Observe  $x_t$  and let for  $a = 1, \dots, K$

$$p_t(a) = (1 - K\rho_{\min}) \frac{\sum_{\pi} \mathbf{1}[\pi(x_t) = a] w_t(\pi)}{\sum_{\pi} w_t(\pi)} + \rho_{\min},$$

where  $\rho_{\min} = \sqrt{\frac{\ln |\Pi|}{KT}}$ .

2. Draw  $a_t$  from  $p_t$ , and observe reward  $r_t(a_t)$ .

3. Update for each  $\pi \in \Pi$

$$w_{t+1}(\pi) = w_t(\pi) \exp \left( \frac{\rho_{\min}}{2} \left( \mathbf{1}[\pi(x_t) = a_t] \frac{r_t(a_t)}{p_t(a_t)} + \frac{1}{p_t(\pi(x_t))} \sqrt{\frac{\ln N/\delta}{KT}} \right) \right)$$

$\hat{y}_i(t)$                        $\hat{v}_i(t)$   
↓                                      ↓

# Lemma 1

The estimated reward of an expert is  $\hat{G}_i \doteq \sum_{t=1}^T \hat{y}_i(t)$ .

We also define  $\hat{\sigma}_i \doteq \sqrt{KT} + \frac{1}{\sqrt{KT}} \sum_{t=1}^T \hat{v}_i(t)$ .

**Lemma**  $\Pr \left[ \exists i : G_i \geq \hat{G}_i + \sqrt{\ln(N/\delta)} \hat{\sigma}_i \right] \leq \delta$ .

Proof uses a new Freedman-style martingale inequality.

# Lemma 2

$$\hat{U} = \max_i \left( \hat{G}_i + \hat{\sigma}_i \cdot \sqrt{\ln(N/\delta)} \right).$$

Lemma

$$G_{\text{Exp4.P}} \geq \left( 1 - 2\sqrt{\frac{K \ln N}{T}} \right) \hat{U} - 2\sqrt{KT \ln(N/\delta)} - \sqrt{KT \ln N} - \ln(N/\delta).$$

Proof tracks the weights of experts, similar to Exp4.

Lemmas 1 and 2 imply :  $G_{\text{Exp4.P}} \geq G_{\text{max}} - 6\sqrt{KT \ln(N/\delta)}$ .

# One Problem...

- This algorithm requires keeping explicit weights on the policies.
  - Okay for polynomially many policies.
  - Okay for some special cases.
  - Not efficient in general.
- Want an efficient algorithm that would (for example) work with an ERM Oracle
  - epoch-greedy [Langford and Zhang '07] has this property.

# Results

Algorithm	Regret	H.P.?	Context?	Efficient?
Exp4 [ACFS '02]	$\tilde{O}(T)^{1/2}$	No	Yes	No
$\epsilon$ -greedy, epoch-greedy [LZ '07]	$\tilde{O}(T^{2/3})$	Yes	Yes	Yes
Exp3.P [ACFS '02] UCB [Auer '00]	$\tilde{O}(T)^{1/2}$	Yes	No	Yes
Exp4.P [BLLRS '10]	$\tilde{O}(T)^{1/2}$	Yes	Yes	No

# Outline

---

- Formally define the setting.
- Show ideas that fail.
- Give a high probability optimal algorithm.
- **Dealing with VC sets.**
- Experiments

# Infinitely Many Policies

- What if we have an infinite number of policies?
- Our bound of  $\tilde{O}(K \ln(N)T)^{1/2}$  becomes vacuous.
- If we assume our policy class has a finite VC dimension  $d$ , then we can tackle this problem.
- Need i.i.d. assumption. We will also assume  $k=2$  to illustrate the argument.

# VC Dimension

- The **VC dimension** of a hypothesis class captures the class's expressive power.
- It is the cardinality of the largest set (in our case, of contexts) the class can shatter.
  - **Shatter** means to label in all possible configurations.

# VE, an Algorithm for VC Sets

The VE algorithm:

- Act uniformly at random for  $\tau$  rounds.
- This partitions our policies  $\Pi$  into equivalence classes according to their labelings of the first  $\tau$  examples.
- Pick one representative from each equivalence class to make  $\Pi'$ .
- Run Exp4.P on  $\Pi'$ .

# Outline of Analysis of VE

- Sauer's lemma bounds the number of equivalence classes to  $(e \tau / d)^d$ .
  - Hence, using Exp4.P bounds, VE's regret to  $\Pi'$  is  $\approx \tau + O(Td \ln(\tau))$
- We can show that the regret of  $\Pi'$  to  $\Pi$  is  $\approx (T/\tau)(d \ln T)$ 
  - by looking at the probability of disagreeing on future data given agreement for  $\tau$  steps.
- $\tau \approx (Td \ln 1/\delta)^{1/2}$  achieves the optimal trade-off.
- Gives  $\tilde{O}(Td)^{1/2}$  regret.
- Still inefficient!

# Outline

---

- Formally define the setting.
- Show ideas that fail.
- Give a high probability optimal algorithm.
- Dealing with VC sets.
- **Experiments**

# Experiments on Yahoo! Data

- We chose a policy class for which we could efficiently keep track of the weights.
  - Created 5 clusters, with users (at each time step) getting features based on their distances to clusters.
  - Policies mapped clusters to article (action) choices.
  - Ran on personalized news article recommendations for Yahoo! front page.
- We used a learning bucket on which we ran the algorithms and a deployment bucket on which we ran the greedy (best) learned policy.

# Experimental Results

- Reported normalized estimated click-through-rates (rewards). Over 41M visits, with 253 articles and 21 candidate articles per visit.

	Exp4.P	Exp4	$\epsilon$ -greedy
learning eCTR	1.0525	1.0988	1.3829
deployment eCTR	<b>1.6512</b>	1.5309	1.4290

# Summary

- Described Exp4P, the first optimal high probability algorithm for the contextual bandit problem.
- Showed how to compete with a VC-Set.
- Experimental Evidence for Exp4.Ps effectiveness.
- Main drawback is efficiency. We only have efficient algorithms for restricted classes, eg. our experiments, linear bandits (Auer 2002, Chu Li R Schapire 2011), etc.
- Main Open Problem: Find an **efficient** optimal algorithm for the contextual bandits problem!
  - Check out John Langford's talk at Snowbird!

# Contextual Bandits in Context



## Supervised Learning

- Limited framework: focus on prediction
- Many tractable algorithms
- Large scale real-world problems

## Reinforcement Learning

- General framework: focus on actions
- Supports stateful problems with limited feedback
- Hard to scale to very large problems

# Contextual Bandits in Context



## Supervised Learning

- Limited framework
- focus on prediction
- Many tractable algorithms
- Large scale problems

## Contextual Bandit Problems

- Optimize for good decisions (not prediction)
- Partial feedback (only see outcome of the action you took)
- Real-world loss functions

## Reinforcement Learning

- focus on decision making
- problems are often large

# Applications

Good for domains with

- Large quantities of data, many rounds
- Decisions must be made in an automated fashion with low latencies --- not possible to have a human in the low-level decision loop

Examples

- Web search advertising
- Content optimization for a news site

# Two general approaches

## 1. Assume structure on the policies for taking actions

Peter Auer. Using confidence bounds for exploitation-exploration tradeoffs.

John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information.

## 2. Assume we can enumerate a set of good policies. Policies are arbitrary functions of context.

Auer et. al. The nonstochastic multiarmed bandit problem.

H. Brendan McMahan and Matthew Streeter. Tighter bounds for multi-armed bandits with expert advice.

# Contributions

- High-probability bounds for EXP4
- *Bounds for infinite policy classes with finite VC dimension*
- *Example of an efficient implementation despite exponentially many experts*
- Experiments

# Experiments

- So far, experimental work on contextual bandits has been secondary to theoretical work.
- Experiments can be tricky:
  - If you only get feedback on the actions you take, how do you evaluate an algorithm that does something different? (Solvable)
  - Parameter tuning issues