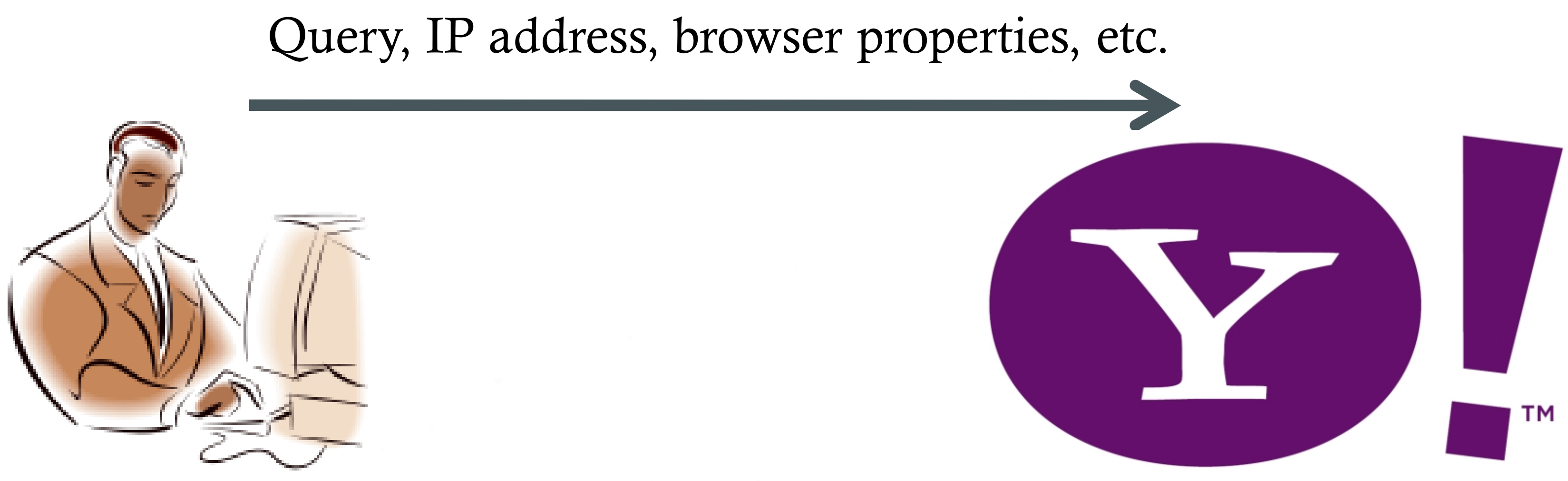
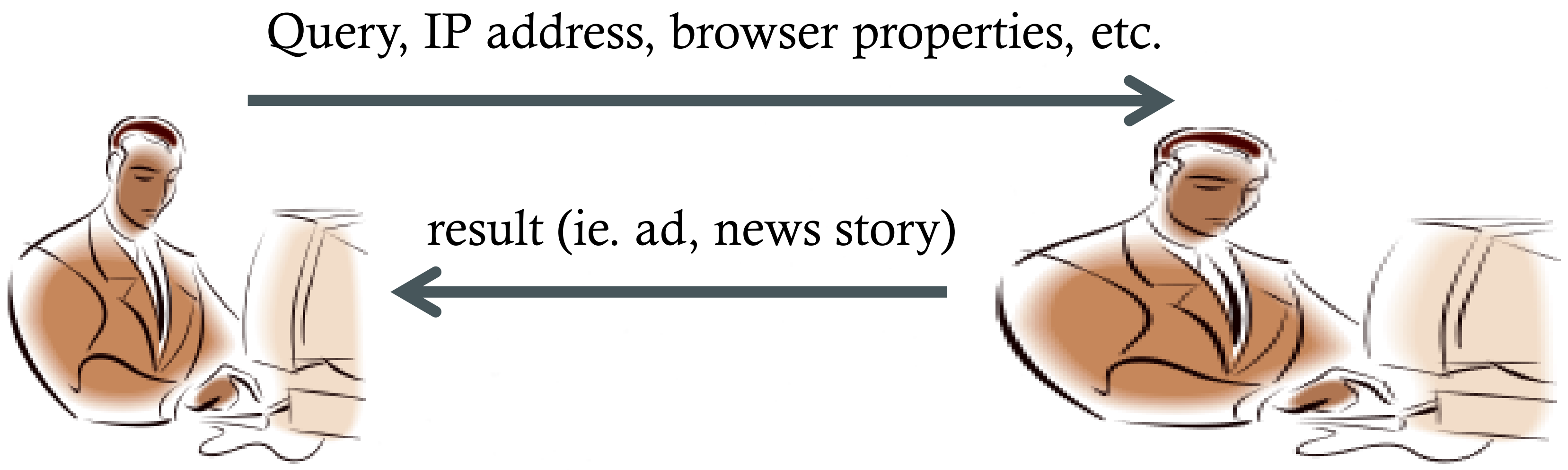
New Algorithms for Contextual Bandits

Lev Reyzin
Yahoo! Research, NY

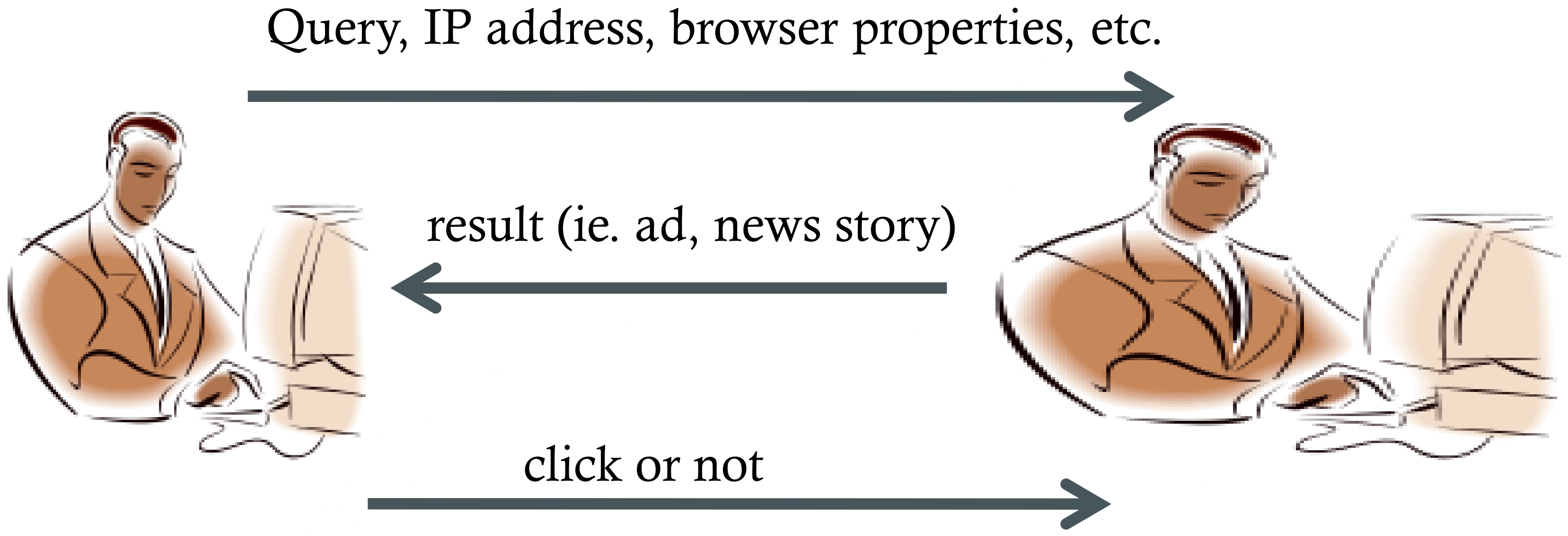
Serving Content to Users



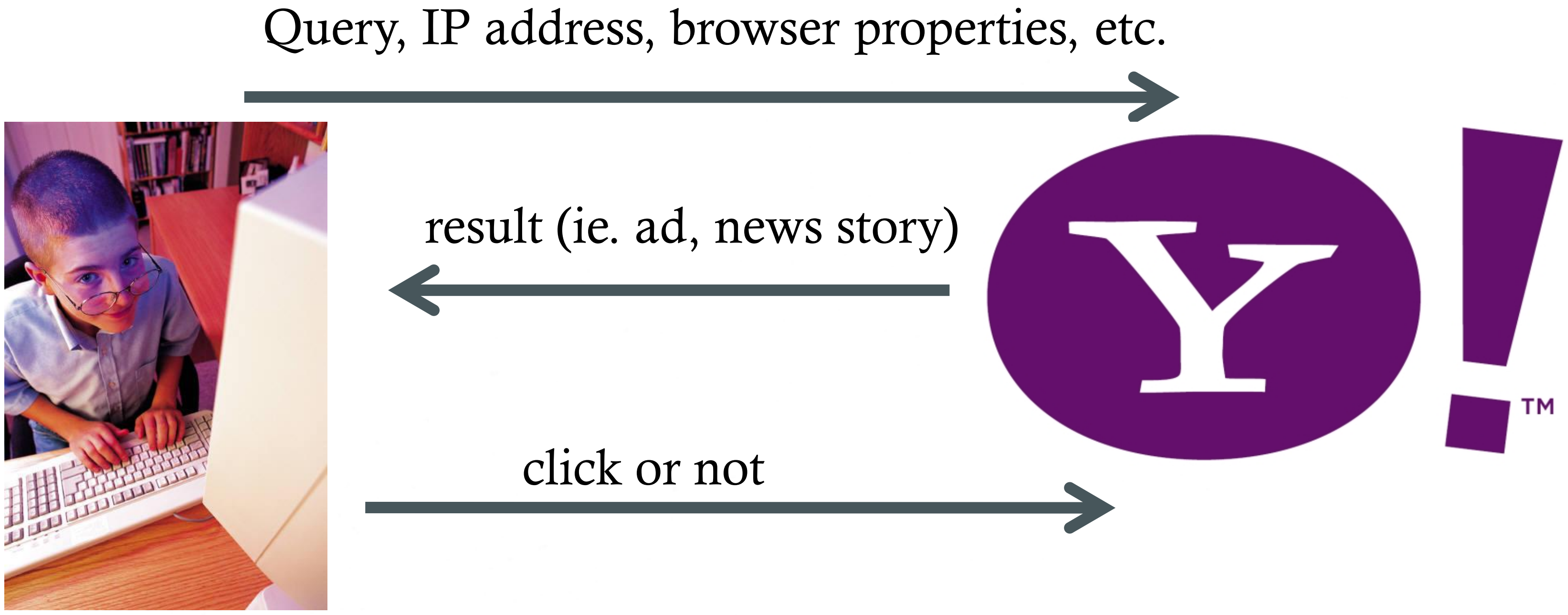
Serving Content to Users


Query, IP address, browser properties, etc.
result (ie. ad, news story)

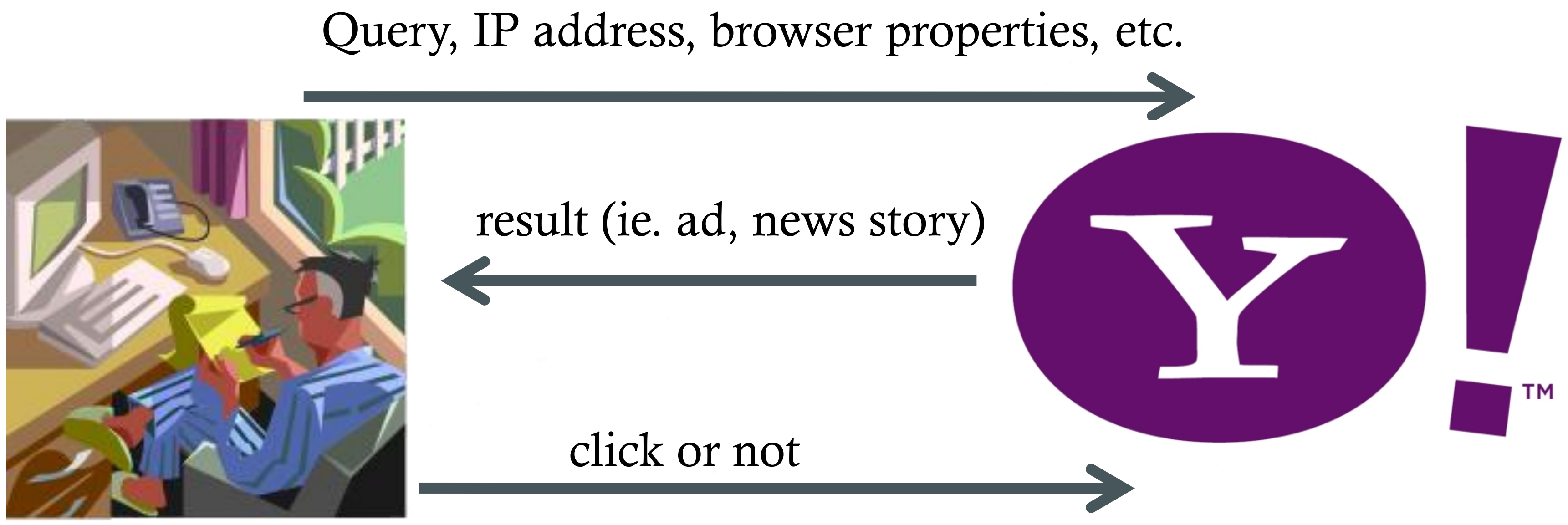
Serving Content to Users


Query, IP address, browser properties, etc.
result (ie. ad, news story)
click or not

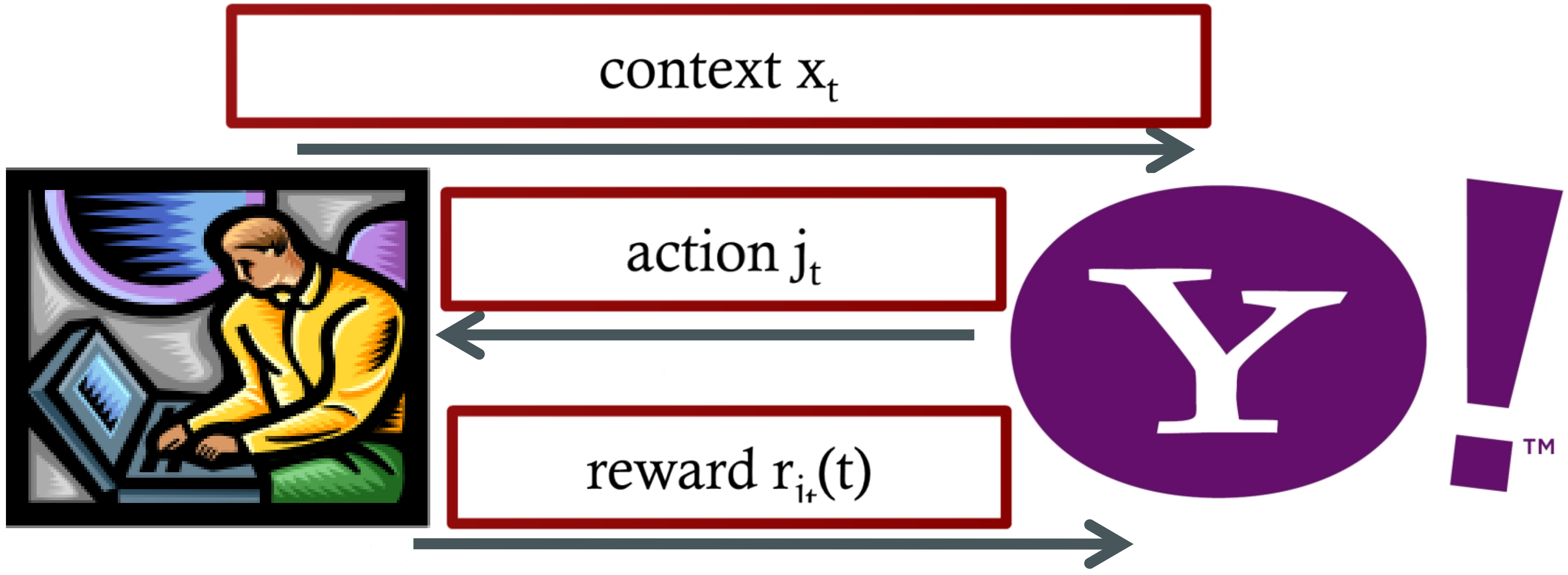
Serving Content to Users


Query, IP address, browser properties, etc.
result (ie. ad, news story)
click or not
Y!
TM

Serving Content to Users


Query, IP address, browser properties, etc.
result (ie. ad, news story)
click or not
Y!
TM

Serving Content to Users



Outline

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

The Setting

- T rounds, K possible actions, N policies π in Π (context → actions)
- for t=1 to T
  - world commits to rewards $\mathbf{r(t)}=r_1(t),r_2(t),\ldots,r_K(t)$
  - world provides context x_t
  - learner's policies recommend $\pi_1(x_t)$, $\pi_2(x_t)$, …, $\pi_N(x_t)$
  - 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: $G_A(T) \triangleq \sum_{t=1}^{T} r_{j_t}(t)$

- expected reward of policy i: $G_i(T) \triangleq \sum_{t=1}^{T} \pi_i(x_t) \cdot r(t)$

- 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 our 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?	Context?
Exp4 [ACFS '02]	$\tilde{O}(KT \ln(N))^{1/2}$	No	Yes
ε-greedy, epoch-geedy [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

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

Our Result

Algorithm	Regret	High Prob?	Context?
Exp4 [ACFS '02]	$\tilde{O}(KT \ln(N))^{1/2}$	No	Yes
ε-greedy, epoch-geedy [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]	$\tilde{O}(K \ln(N/\delta)T)^{1/2}$	Yes	Yes

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

Outline

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

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 ~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 ~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 ~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 > ½ it is never picked and algorithm incurs regret of T.

epsilon-greedy

- Rough idea of ε-greedy (or ε-first): act randomly for ε rounds, then go with best (arm or expert).
- 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: $\varepsilon+(T-\varepsilon)/(\varepsilon^{1/2})$.
  - $\varepsilon= T^{2/3}$ is optimal.
  - gives regret $T^{2/3}$

Outline

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

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 $t = 1, 2, \ldots$:

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

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

   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 $t = 1, 2, \ldots$:

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

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

   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 $\pi \in \Pi$

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

Why Should This Work?

$$w_i(t+1) = w_i(t)\exp\left(\frac{\sqrt{\ln N}}{2\sqrt{KT}}\left(\hat{g}_i(t) + \hat{v}_i(t)\sqrt{\frac{\ln(N/\delta)}{KT}}\right)\right)$$

Why Should This Work?

$$w_i(t+1) \;=\; w_i(t)\exp\left(\frac{\sqrt{\ln N}}{2\sqrt{KT}}\underbrace{\left(\hat{y}_i(t)+\hat{v}_i(t)\sqrt{\frac{\ln(N/\delta)}{KT}}\right)}\right)$$

$$\sim \sum_i^T \hat{y}_i(t) + \sqrt{\ln(N/\delta)/KT}\sum_i^T \hat{v}_i(t)$$

$$\sim \hat{G}_i + \sqrt{\ln(N/\delta)}\hat{\sigma}_i$$

$\geq G_i$ w.p. $\geq 1-\delta$

using a martingale inequality

Proof Outline

so we have $\hat{G}_i + \sqrt{\ln(N/\delta)}\hat{\sigma}_i \geq G_i$ w.p. $\geq 1 - \delta$

Proof Outline

so we have $\hat{G}_i + \sqrt{\ln(N/\delta)}\hat{\sigma}_i \geq G_i$ w.p. $\geq 1 - \delta$

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

by looking at $\ln(W_{T+1}/W_1)$

we can show

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

Proof Outline

so we have $\hat{G}_i + \sqrt{\ln(N/\delta)}\hat{\sigma}_i \geq G_i$ w.p. $\geq 1 - \delta$

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

Proof Outline

so we have $\hat{G}_i + \sqrt{\ln(N/\delta)}\hat{\sigma}_i \geq G_i$ w.p. $\geq 1 - \delta$

and

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

implies $G_{\text{Exp4.P}} \geq G_{\max} - O\left(\sqrt{KT \ln(N/\delta)}\right)$ w.p. $1 - \delta$

Exp4P beats epslion-greedy in practice [BLLRS '10] and performs negligibly worse (on average) than Exp4.

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
ε-greedy, epoch-geedy [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.**
- Linear Payoff Functions.
- Slates.

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 [Beygelzimer, Langford, Li, R, Schapire '10] :

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

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 Π' is $\approx \tau + O(Td \ln(\tau))$
- We can show that the regret of Π' to Π is $\approx (T/\tau)(d\ln T)$
  - by looking at the probability of disagreeing on future data given agreement for τ 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.
- **Linear Payoff Functions.**
- Slates.

When Do Efficient Algorithms Exist?

- One example is the linear payoff functions setting.
- In this setting, on each round, we observe a d dimensional context (feature vector) for each arm.
    - Equivalent to the contextual bandit setting.
- We assume there exists an unknown vector, whose dot product with each arm feature gives the expected regret of that arm.
    - Similar to a realizability assumption.

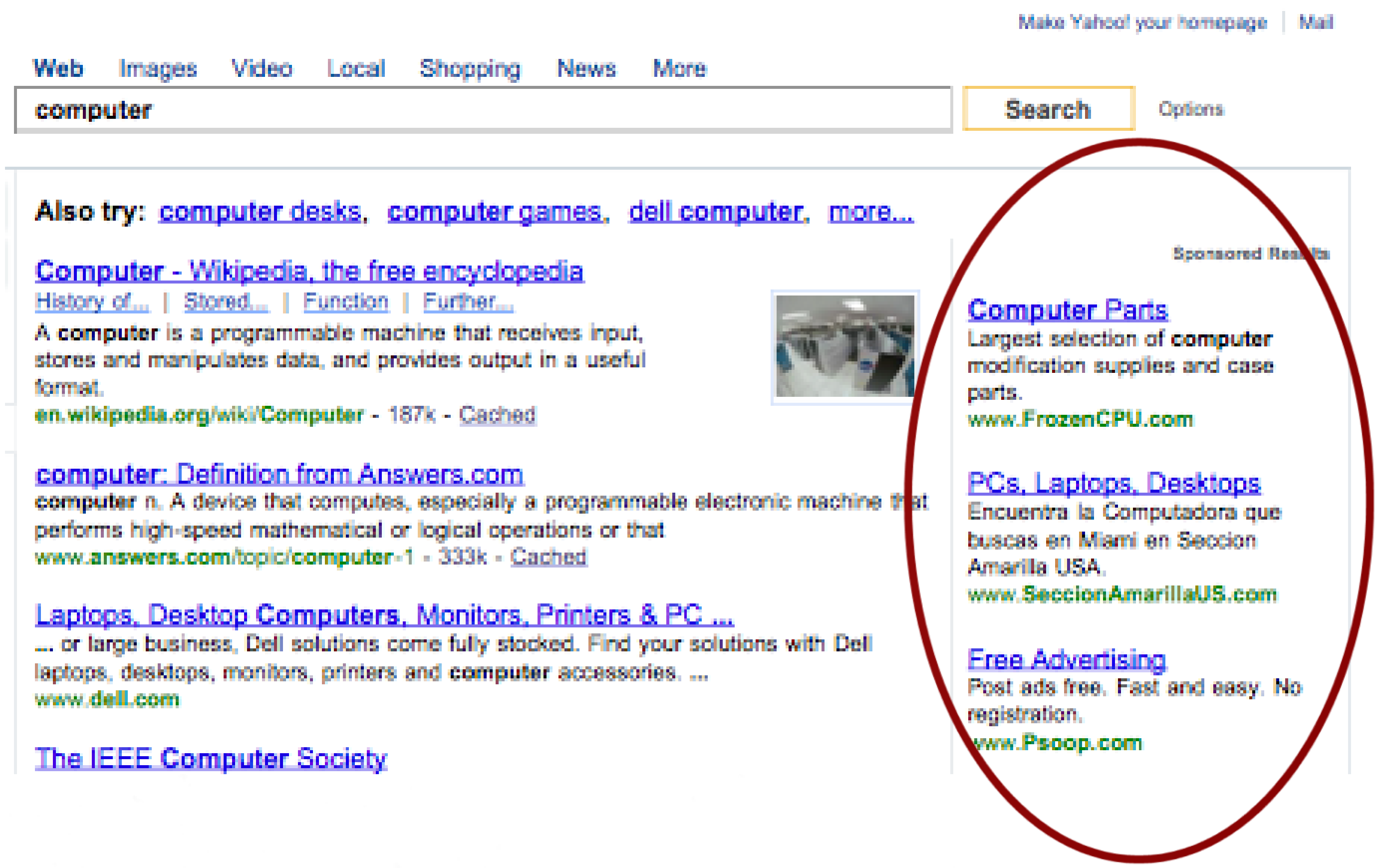
Linear Payoffs

- LinRel [Auer '02] gives a polynomial time algorithm with $\tilde{O}(Td)^{1/2}$ regret.
- LinRel tries to estimate the reward of the current round by looking at past rounds.
  - LinRel decomposes the feature vector of the current round into a linear combination of feature vectors seen on previous rounds.
  - Looks at previous rewards to compute coefficients.
  - Uses these estimates to compute reward estimates.
- Matches the $\Omega(Td)^{1/2}$ lower bound [Chu, Li, R, Schapire '10] up to log factors.
- LinUCB (a similar algorithm to LinRel) outperforms epsilon-greedy on Yahoo! Homepage data. [Li, Chu, Langford, Schapire '10].

Outline

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

Slates



Make Yahoo! your homepage | Mail

Web Images Video Local Shopping News More

computer

Search Options

Also try: computer desks, computer games, dell computer, more...

Computer - Wikipedia, the free encyclopedia
History of... | Stored... | Function | Further...
A **computer** is a programmable machine that receives input, stores and manipulates data, and provides output in a useful format.
en.wikipedia.org/wiki/Computer - 187k - Cached

computer: Definition from Answers.com
computer n. A device that computes, especially a programmable electronic machine that performs high-speed mathematical or logical operations or that
www.answers.com/topic/computer-1 - 333k - Cached

Laptops, Desktop Computers, Monitors, Printers & PC ...
... or large business, Dell solutions come fully stocked. Find your solutions with Dell laptops, desktops, monitors, printers and **computer** accessories. ...
www.dell.com

The IEEE Computer Society

Sponsored Results

Computer Parts
Largest selection of **computer** modification supplies and case parts.
www.FrozenCPU.com

PCs, Laptops, Desktops
Encuentra la Computadora que buscas en Miami en Seccion Amarilla USA.
www.SeccionAmarillaUS.com

Free Advertising
Post ads free. Fast and easy. No registration.
www.Psoop.com

Slates

- Oftentimes, we have to choose multiple actions (without replacement).
- Different models:
  - All slots equal.
  - Positional factors.
  - Interaction via "properties."

Slates

- For T time steps, K actions, s slots, N experts choosing slates, we can get the following regret bounds [Kale, R, Schapire '10]:
  - $O((sKT\ln N)^{1/2})$ for unordered slates
  - $O(s(KT\ln N)^{1/2})$ for ordered slates
- Beats a straightforward reduction to Exp4.
- Uses a variant of multiplicative weights.
- Not efficient.

Summary

- Described Exp4P, the first optimal high probability algorithm for the contextual bandit problem.
- Showed how to compete with a VC-Set.
- Discussed an efficient linear-payoff algorithm.
- Introduced the slates problem.

Open Problems

- Main Open Problem: Find an efficient optimal algorithm for the contextual bandits problem!
- i.e. make the Exp4P algorithm efficient with an ERM (Empirical Risk Minimization) oracle.
  - Instead of updating weights explicitly for each policy, feed rewards for actions into an oracle, which can return a good policy.
  - This oracle could be a standard efficient learning algorithm.

Open Problems

- Find good classes of policies for contextual bandit problems. Linear policies seem to do well…
- Get rid of the realizability assumption in LinRel or LinUCB for linear payoffs.
- Deal with interaction in slates!
- More experimental evaluation.