New Algorithms for Contextual Bandits

Lev Reyzin
Georgia Institute of Technology

Work done at Yahoo!

- A. Beygelzimer, J. Langford, L. Li, L. Reyzin, R.E. Schapire Contextual Bandit Algorithms with Supervised Learning Guarantees (AISTATS 2011)

- M. Dudik, D. Hsu, S. Kale, N. Karampatziakis, J. Langford, L. Reyzin, T. Zhang Efficient Optimal Learning for Contextual Bandits (UAI 2011)

- S. Kale, L. Reyzin, R.E. Schapire Non-Stochastic Bandit Slate Problems (NIPS 2010)

Serving Content to Users

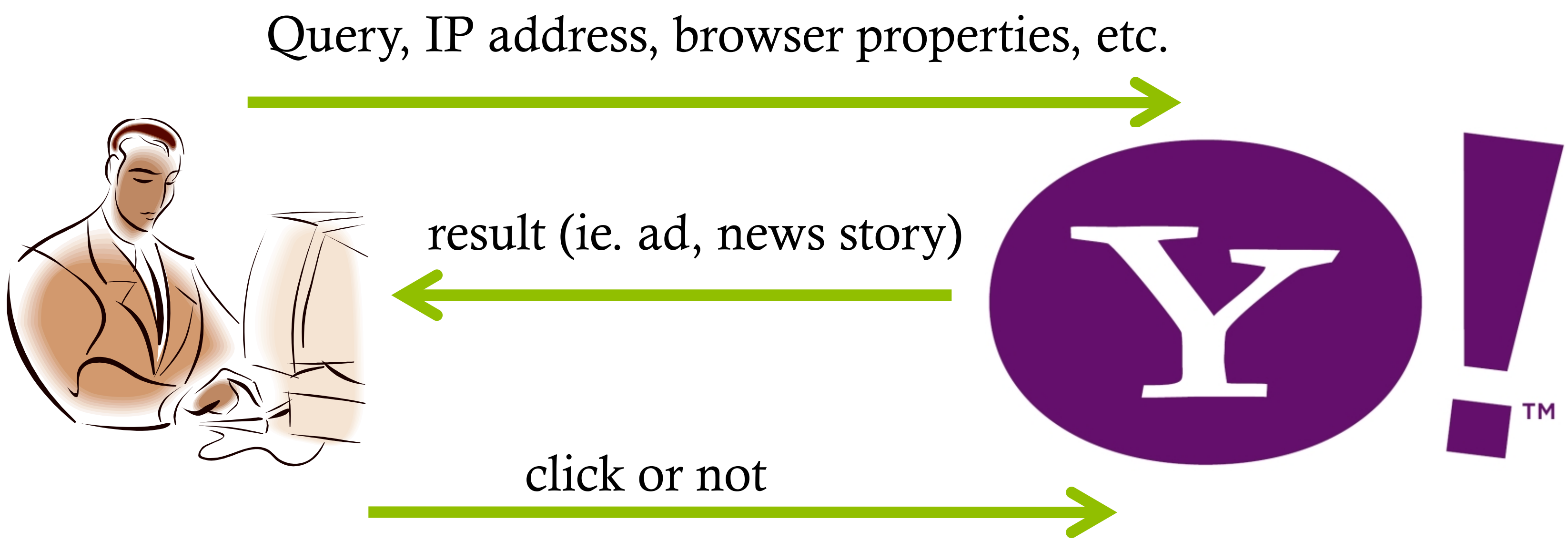
Query, IP address, browser properties, etc.

Serving Content to Users

Query, IP address, browser properties, etc.

result (ie. ad, news story)

Serving Content to Users



Serving Content to Users

Query, IP address, browser properties, etc.

result (ie. ad, news story)

click or not

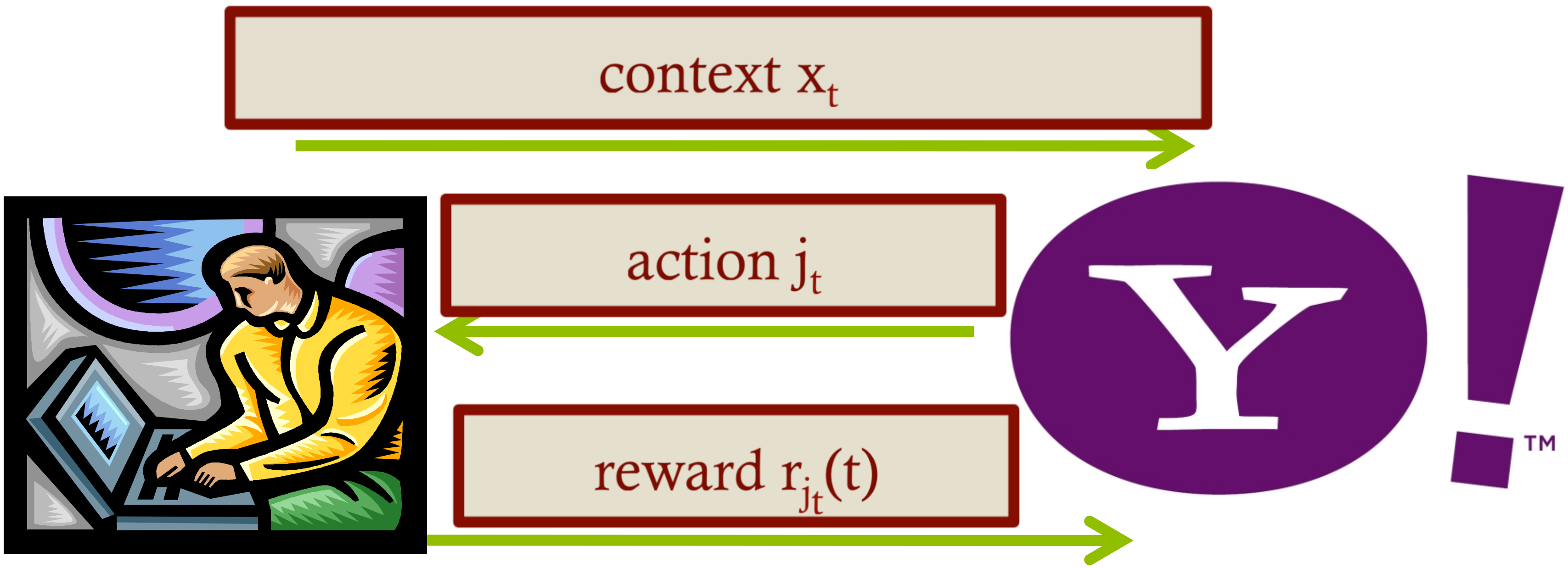
Serving Content to Users

Query, IP address, browser properties, etc.

result (ie. ad, news story)

click or not

Serving Content to Users



Outline

- **The setting and some background**
- Show ideas that fail
- Give a high probability optimal algorithm
- Dealing with VC sets
- An efficient algorithm
- Slates

Multiarmed Bandits

[Robbins ’52]

1 2 3 4 … T

1 

2 

3 

⋮

k 

Multiarmed Bandits

[Robbins ’52]

1 2 3 4 … T

1 

2  $0.50

3 

⋮

k 

Multiarmed Bandits

[Robbins '52]

1 2 3 4 … T

1 

2  $0.50

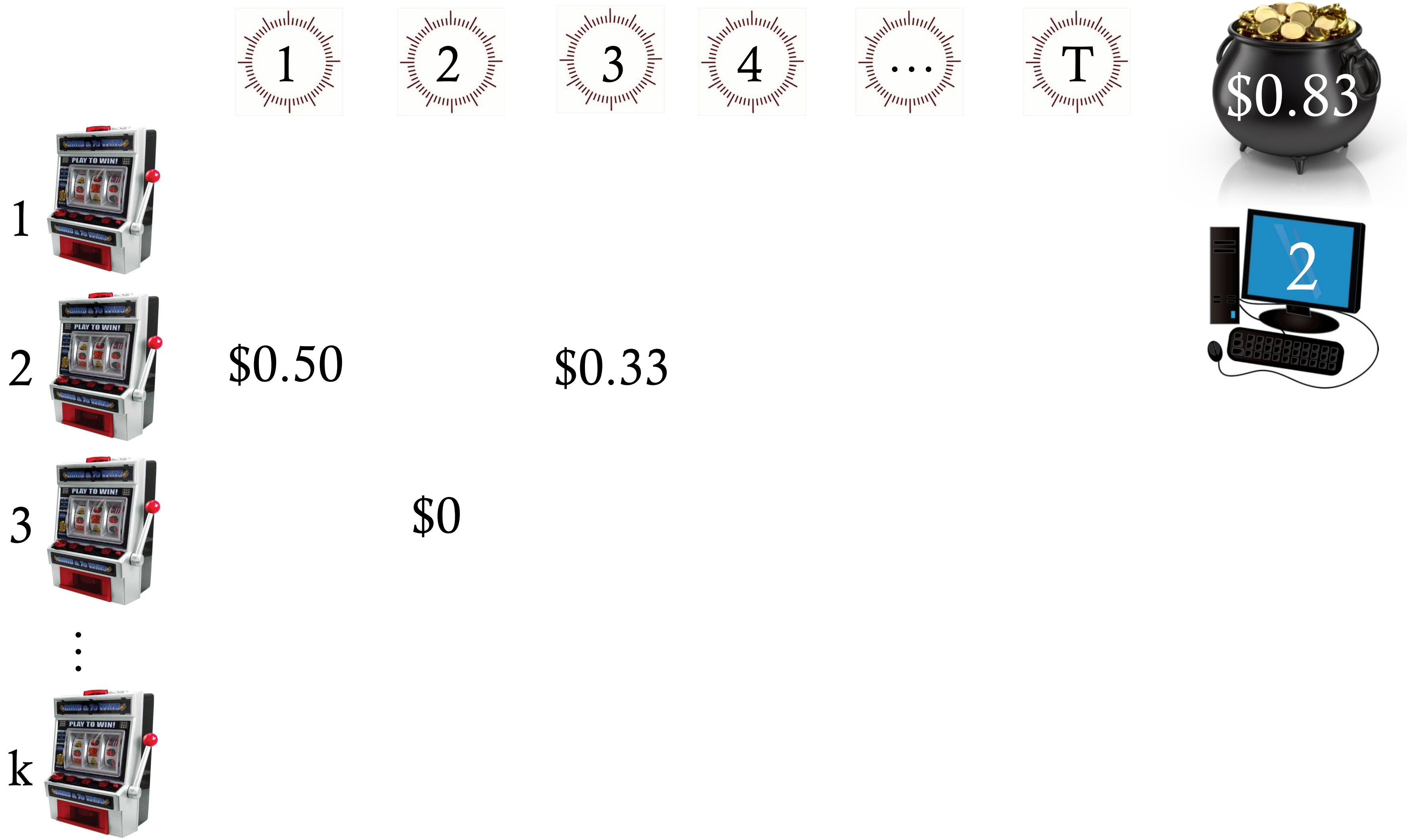
3  $0

⋮

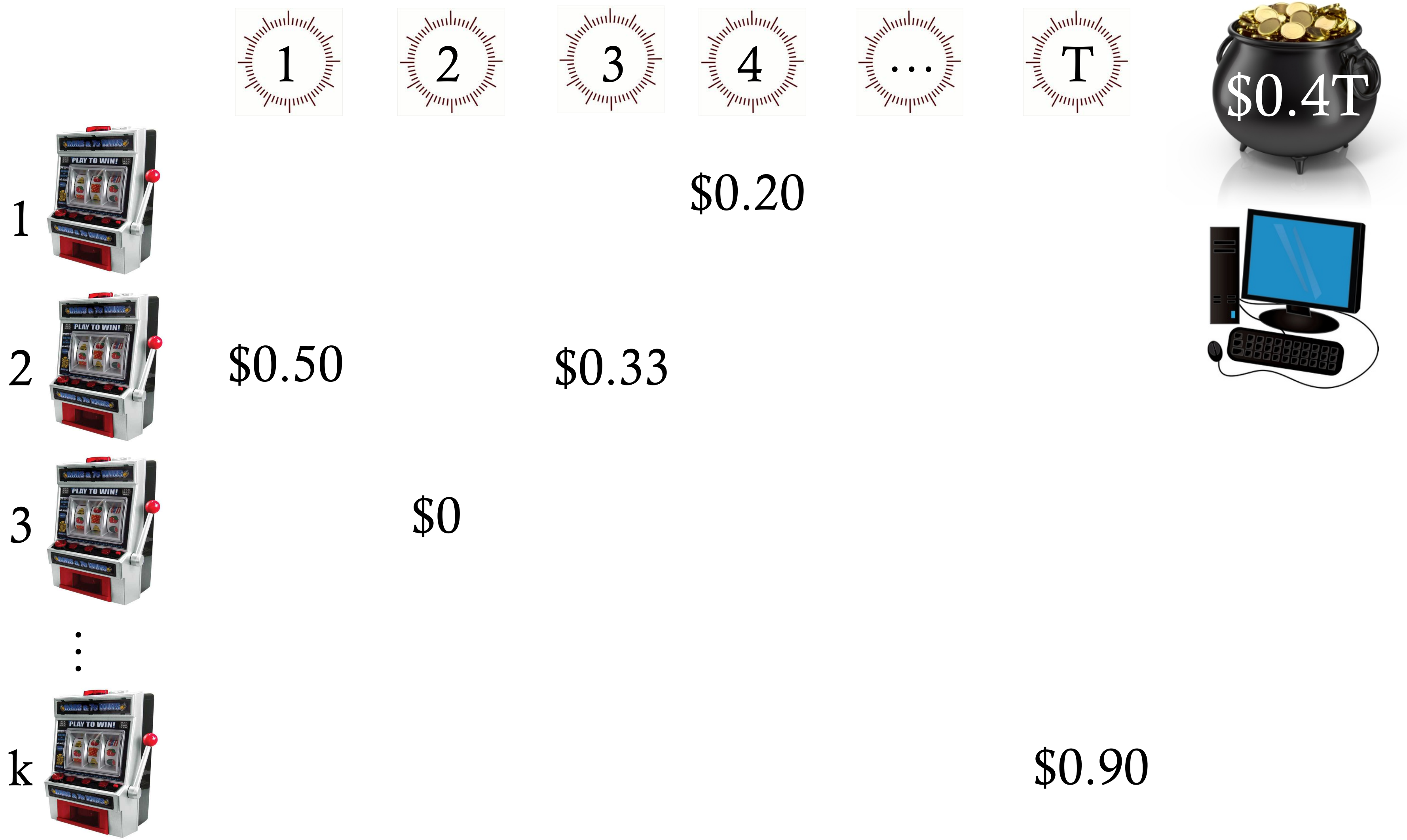
k 

Multiarmed Bandits

[Robbins ’52]



Multiarmed Bandits
[Robbins ’52]



Multiarmed Bandits

[Robbins ’52]

1  $\longrightarrow$ \$0.5T

2  $\longrightarrow$ \$0.2T

3  $\longrightarrow$ \$0.33T

$\vdots$

k  $\longrightarrow$ \$0.1T

Multiarmed Bandits

[Robbins ’52]

1  **$0.5T**

2  $0.2T

3  $0.33T

⋮

k  $0.1T

regret = $0.1T

Contextual Bandits

[Auer-CesaBianchi-Freund-Schapire '02]

context:

1 2 3 4 … T

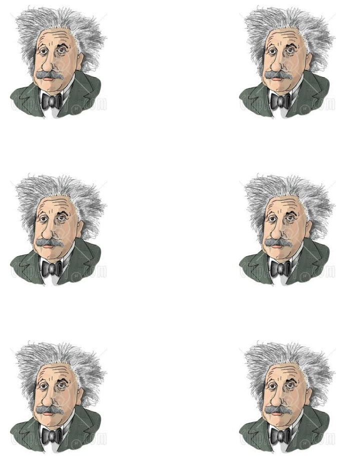
1 

2 

3 

⋮

k 



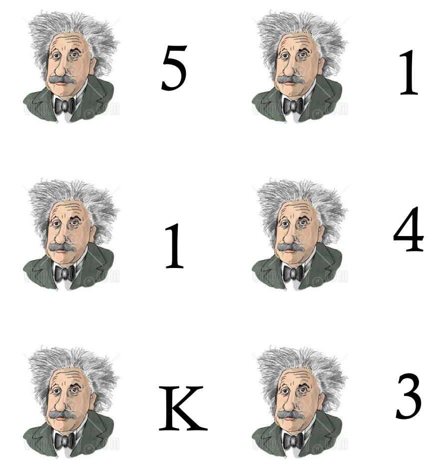
N experts/policies/functions
think of N >> K

Contextual Bandits

[Auer-CesaBianchi-Freund-Schapire '02]

context: x_1

1 2 3 4 … T



N experts/policies/functions
think of N >> K

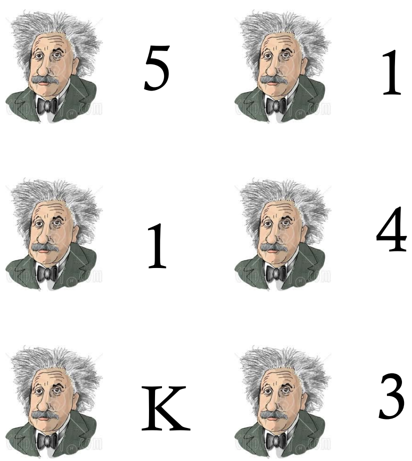
Contextual Bandits

[Auer-CesaBianchi-Freund-Schapire ’02]

context: x_1

1 2 3 4 … T

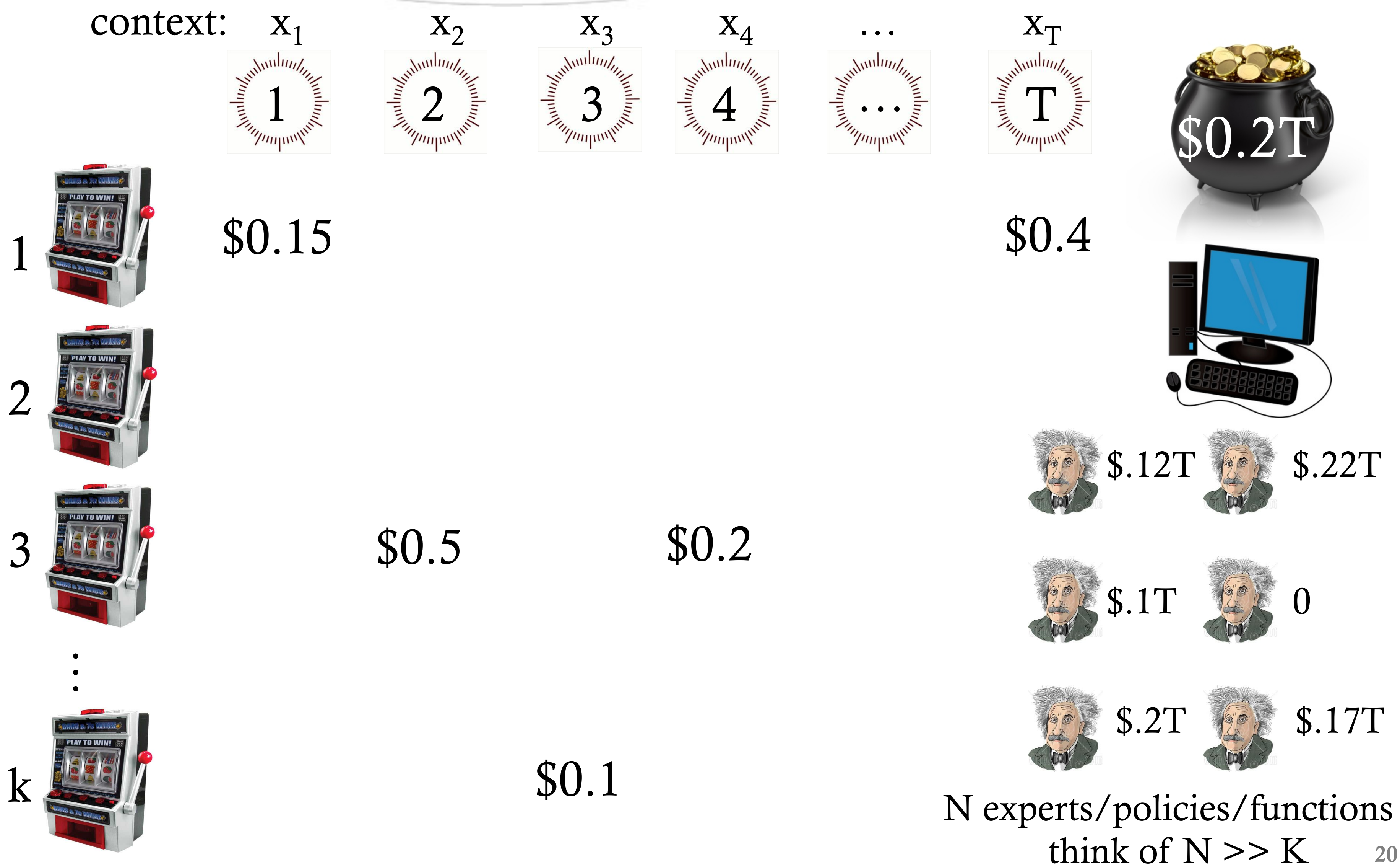
$0.15



N experts/policies/functions
think of N >> K

Contextual Bandits

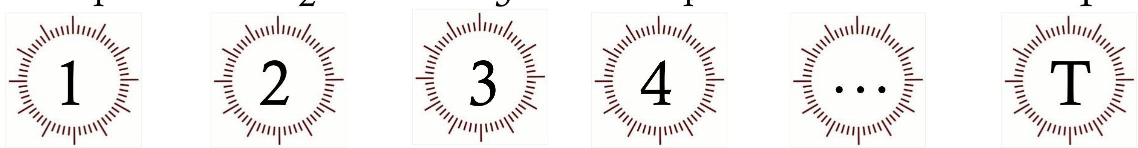
[Auer-CesaBianchi-Freund-Schapire '02]



Contextual Bandits

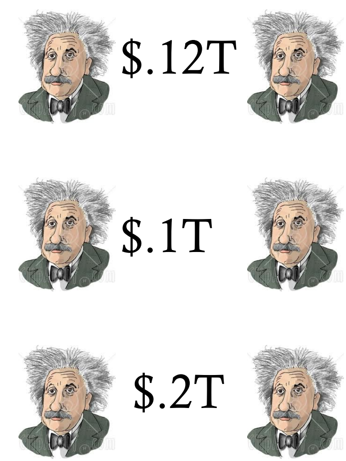
[Auer-CesaBianchi-Freund-Schapire ’02]

context: x_1 x_2 x_3 x_4 … x_T



1
2
3
⋮
k

regret = \$0.02T



\$.12T \$.22T

\$.1T 0

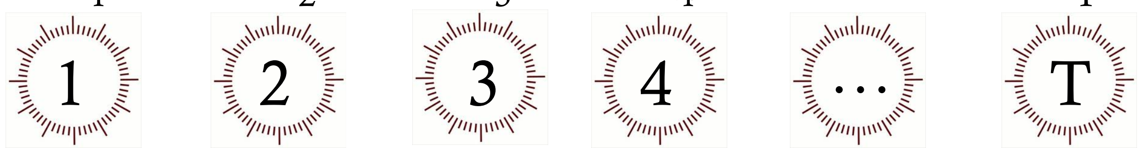
\$.2T \$.17T

N experts/policies/functions

think of N >> K

Contextual Bandits
[Auer-CesaBianchi-Freund-Schapire '02]

context: x_1 x_2 x_3 x_4 … x_T

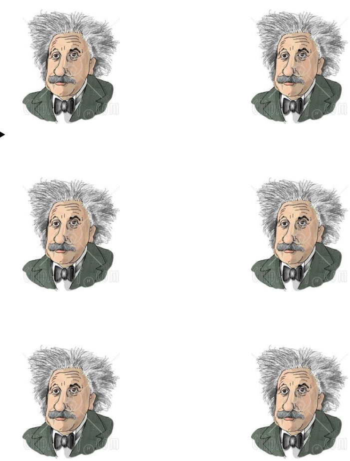


1
2
3
⋮
k

←

the rewards can come i.i.d. from a distribution or be arbitrary
stochastic / adversarial

The experts can be present or not. →
contextual / non-contextual



The Setting

- T rounds, K possible actions, N policies π in Π (context → actions)
- for t=1 to T
  - world commits to rewards **r(t)**=$r_1(t), r_2(t), \ldots, r_K(t)$ (adversarial or iid)
  - world provides context x_t
  - learner's policies recommend $\pi_1(x_t), \pi_2(x_t), \ldots, \pi_N(x_t)$
  - 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: $\max_i G_i(T) - G_A(T)$
- bound on expected regret: $\max_i G_i(T) - E[G_A(T)] < \varepsilon$
- high probability bound: $P[\max_i G_i(T) - G_A(T) > \varepsilon] \leq \delta$

- Harder than supervised learning:
  - In the bandit setting we do not know the rewards of actions not taken.
- Many applications
  - Ad auctions, medicine, finance, …
- Exploration/Exploitation
  - Can exploit expert/arm you've learned to be good.
  - Can explore expert/arm you're not sure about.

Some Barriers

$\Omega(kT)^{1/2}$ (non-contextual) and ~ $\Omega(TK \ln N)^{1/2}$ (contextual) are known lower bounds [Auer et al. '02] on regret, even in the stochastic case.

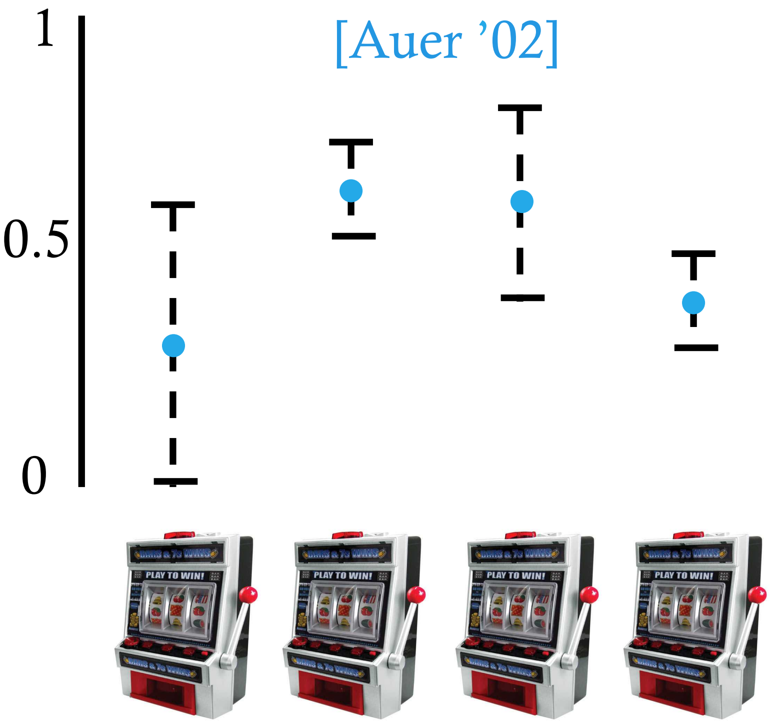
Any algorithm achieving regret $\tilde{O}(KT \text{ polylog } N)^{1/2}$ is said to be optimal.

ε -greedy algorithms that first explore (act randomly) and then exploit (follow the best policy) cannot be optimal. Any optimal algorithm must be adaptive.

Two Types of Approaches

UCB

[Auer '02]

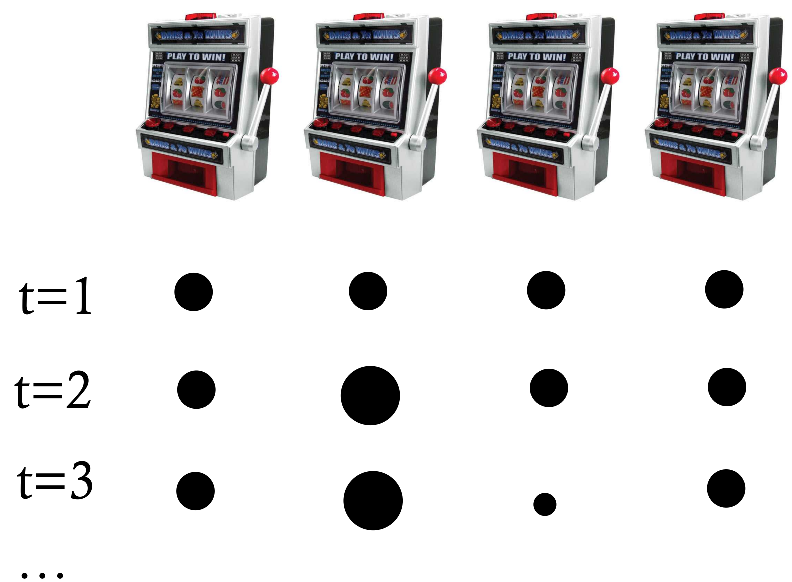


Algorithm: at every time step

1) pull arm with highest UCB
2) update confidence bound of the arm pulled.

EXP3 Exponential Weights

[Littlestone-Warmuth '94]
[Auer et al. '02]



Algorithm: at every time step

1) sample from distribution defined by weights (mixed w/ uniform)
2) update weights "exponentially"

UCB vs EXP3
A Comparison

UCB
[Auer '02]

- Pros
  - Optimal for the stochastic setting.
  - Succeeds with high probability.
- Cons
  - Does not work in the adversarial setting.
  - Is not optimal in the contextual setting.

EXP3 & Friends
[Auer-CesaBianchi-Freund-Schapire '02]

- Pros
  - Optimal for both the adversarial and stochastic settings.
  - Can be made to work in the contextual setting
- Cons
  - Does not succeed with high probability in the contextual setting (only in expectation).

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]

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]

EXP4P
[Beygelzimer-Langford-Li-**R**-Schapire '11]

Main Theorem [Beygelzimer-Langford-Li-**R**-Schapire '11]: For any $\delta > 0$, with probability at least $1-\delta$, EXP4P has regret at most $O(KT \ln (N/\delta))^{1/2}$ in the adversarial contextual bandit setting.

EXP4P combines the advantages of Exponential Weights and UCB.

optimal for both the stochastic and adversarial settings

works for the contextual case (and also the non-contextual case)

a high probability result

Outline

- The setting and some background
- **Show ideas that fail**
- Give a high probability optimal algorithm
- Dealing with VC sets
- An efficient algorithm
- Slates

Some Failed Approaches

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

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.

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.

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, ε-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 of $\varepsilon+(T-\varepsilon)/(\varepsilon^{1/2})$.
  - $\varepsilon \approx T^{2/3}$ is optimal tradeoff.
  - gives regret $\approx T^{2/3}$

Outline

- The setting and some background
- Show ideas that fail
- **Give a high probability optimal algorithm**
- Dealing with VC sets
- An efficient algorithm
- Slates

Ideas Behind Exp4.P

(all appeared in previous algorithms)

- 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 $a = 1, \ldots, K$

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

   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 $\pi \in \Pi$

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

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 $a = 1, \ldots, K$

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

   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 $\pi \in \Pi$

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

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

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.

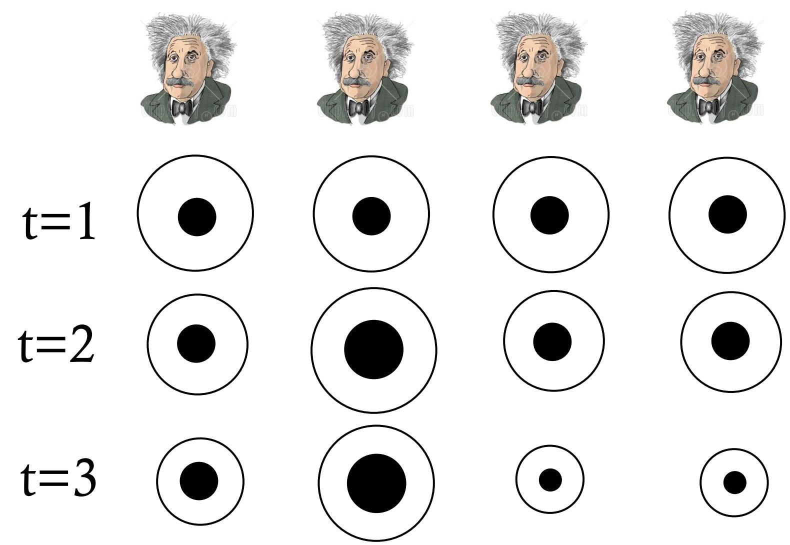
Lemmas 1 and 2 imply : $G_{\text{Exp4.P}} \geq G_{\max} - 6\sqrt{KT \ln(N/\delta)}.$

EXP4P
[Beygelzimer-Langford-Li-**R**-Schapire '11]

Main Theorem [Beygelzimer-Langford-Li-**R**-Schapire '11]: For any $\delta > 0$, with probability at least $1-\delta$, EXP4P has regret at most $O(KT \ln (N/\delta))^{1/2}$ in the adversarial contextual bandit setting.

key insights (on top of UCB/ EXP)

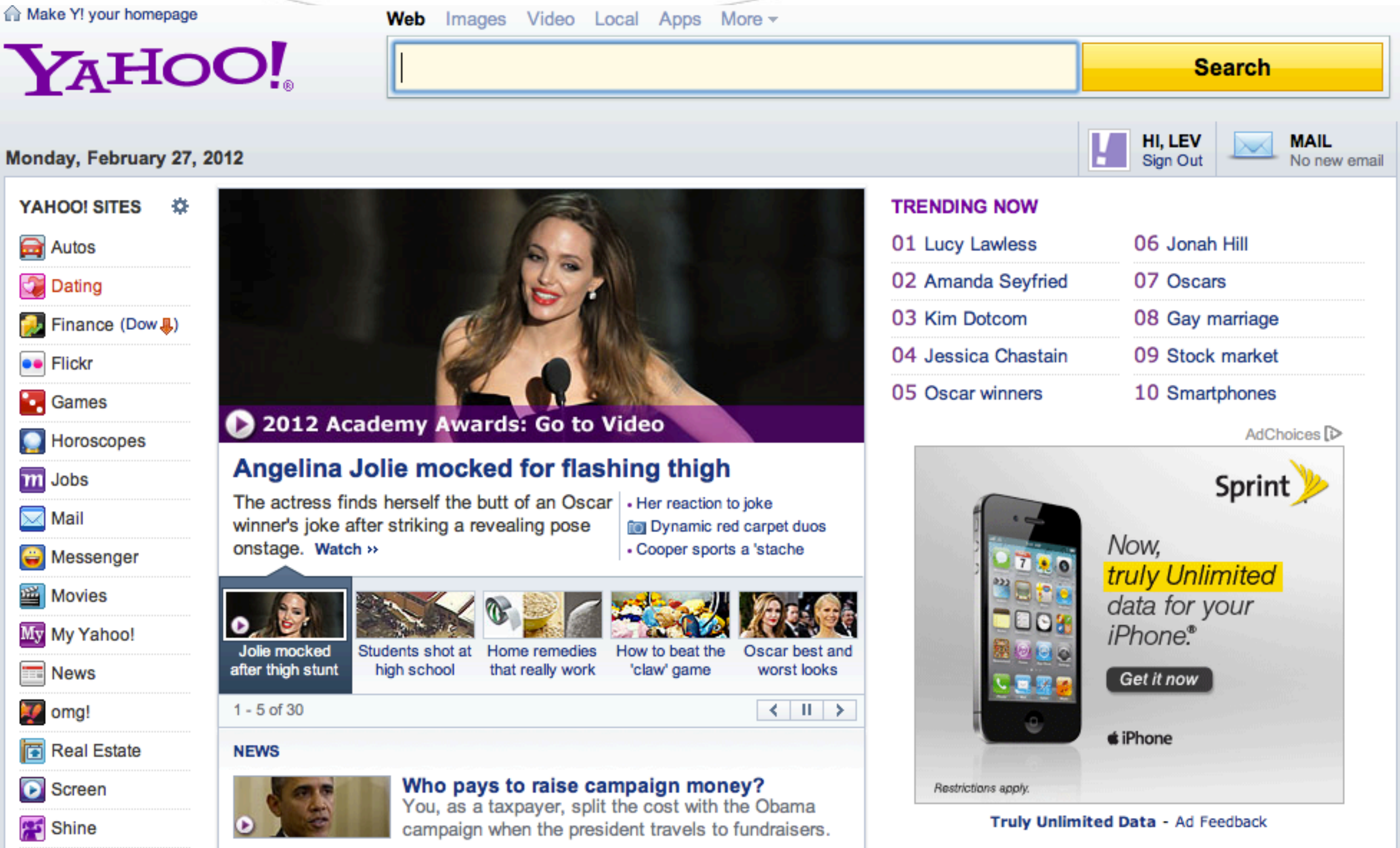
1) exponential weights and upper confidence bounds "stack"
2) generalized Bernstein's inequality for martingales



Efficiency

Algorithm	Regret	High Prob?	Context?	Efficient?
Exp4 [ACFS '02]	$\tilde{O}(T^{1/2})$	No	Yes	No
epoch-geedy [LZ '07]	$\tilde{O}(T^{2/3})$	Yes	Yes	Yes
Exp3.P/UCB [ACFS '02][A '00]	$\tilde{O}(T^{1/2})$	Yes	No	Yes
Exp4.P [BLLRS '10]	$\tilde{O}(T^{1/2})$	Yes	Yes	**No**

EXP4P Applied to Yahoo!



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 estimated (normalized) click-through rates on front page news. Over 41M user visits. 253 total articles. 21 candidate articles per visit.

	EXP4P	EXP4	ε-greedy
Learning eCTR	1.0525	1.0988	1.3829
Deployment eCTR	**1.6512**	1.5309	1.4290

Experimental Results

Reported estimated (normalized) click-through rates on front page news. Over 41M user visits. 253 total articles. 21 candidate articles per visit.

	EXP4P	EXP4	ε-greedy
Learning eCTR	1.0525	1.0988	1.3829
Deployment eCTR	**1.6512**	1.5309	1.4290

Why does this work in practice?

Outline

- The setting and some background
- Show ideas that fail
- Give a high probability optimal algorithm
- **Dealing with VC sets**
- An efficient algorithm
- 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:

- 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

- The setting and some background
- Show ideas that fail
- Give a high probability optimal algorithm
- Dealing with VC sets
- **An efficient algorithm**
- Slates

Hope for an Efficient Algorithm?

[Dudik-Hsu-Kale-Karampatziakis-Langford-**R**-Zhang '11]

For EXP4P, the dependence on N in the regret is logarithmic.

this suggests

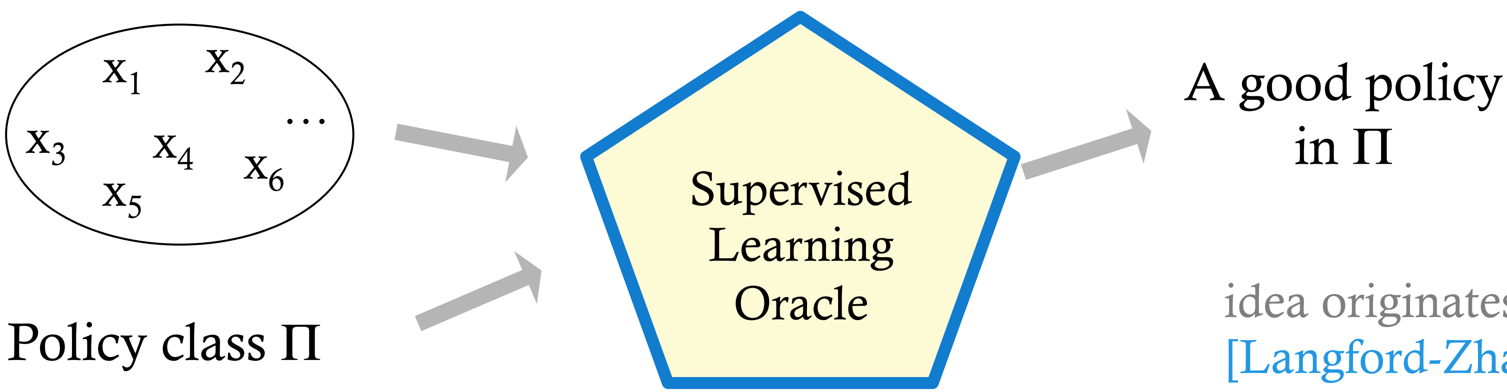
We could compete with a large, even super-polynomial number of policies! (e.g. $N=K^{100}$ becomes $10 \log^{1/2} K$ in the regret)

however

All known contextual bandit algorithms explicitly "keep track" of the N policies. Even worse, just reading in the N would take too long for large N.

Idea: Use Supervised Learning

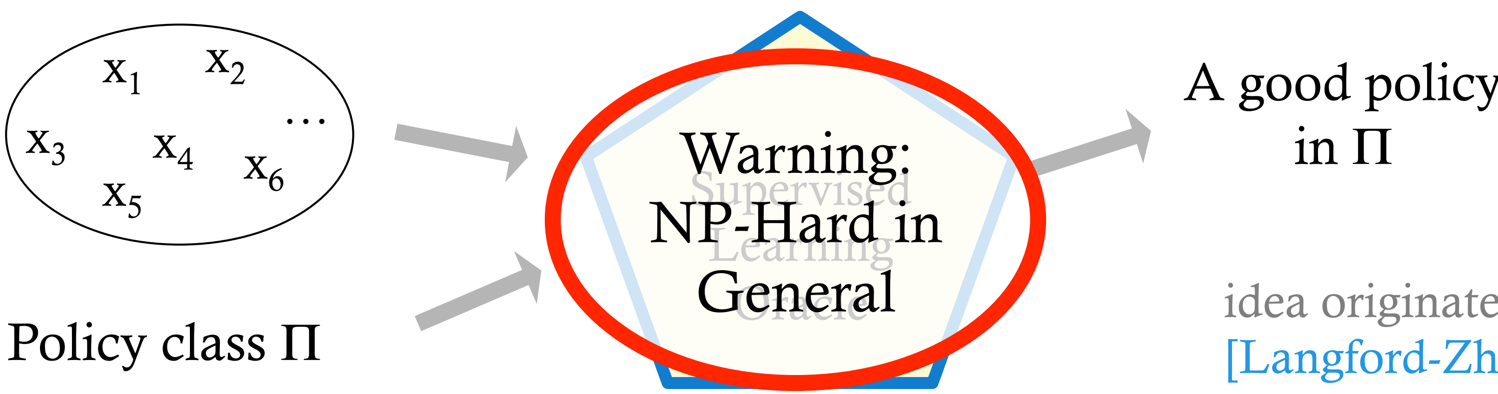
- "Competing" with a large (even exponentially large) set of policies is commonplace in supervised learning.
  - Targets: e.g. linear thresholds, CNF, decision trees (in practice only)
  - Methods: e.g. boosting, SVM, neural networks, gradient descent
- The recommendations of the policies don't need to be explicitly read in when the policy class has structure!



idea originates with [Langford-Zhang '07]

Idea: Use Supervised Learning

- "Competing" with a large (even exponentially large) set of policies is commonplace in supervised learning.
  - Targets: e.g. linear thresholds, CNF, decision trees (in practice only)
  - Methods: e.g. boosting, SVM, neural networks, gradient descent
- The recommendations of the policies don't need to be explicitly read in when the policy class has structure!

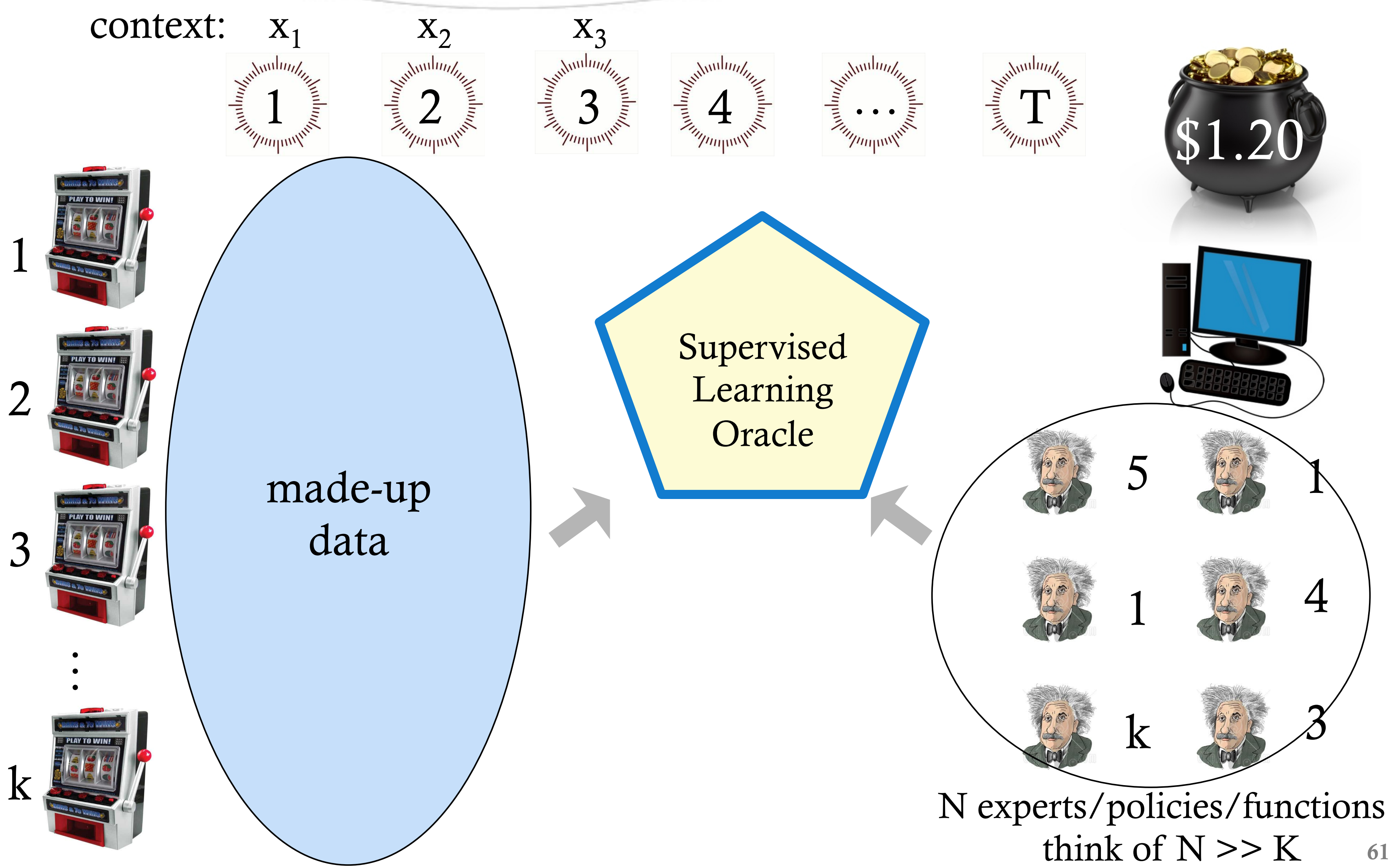


idea originates with [Langford-Zhang '07]

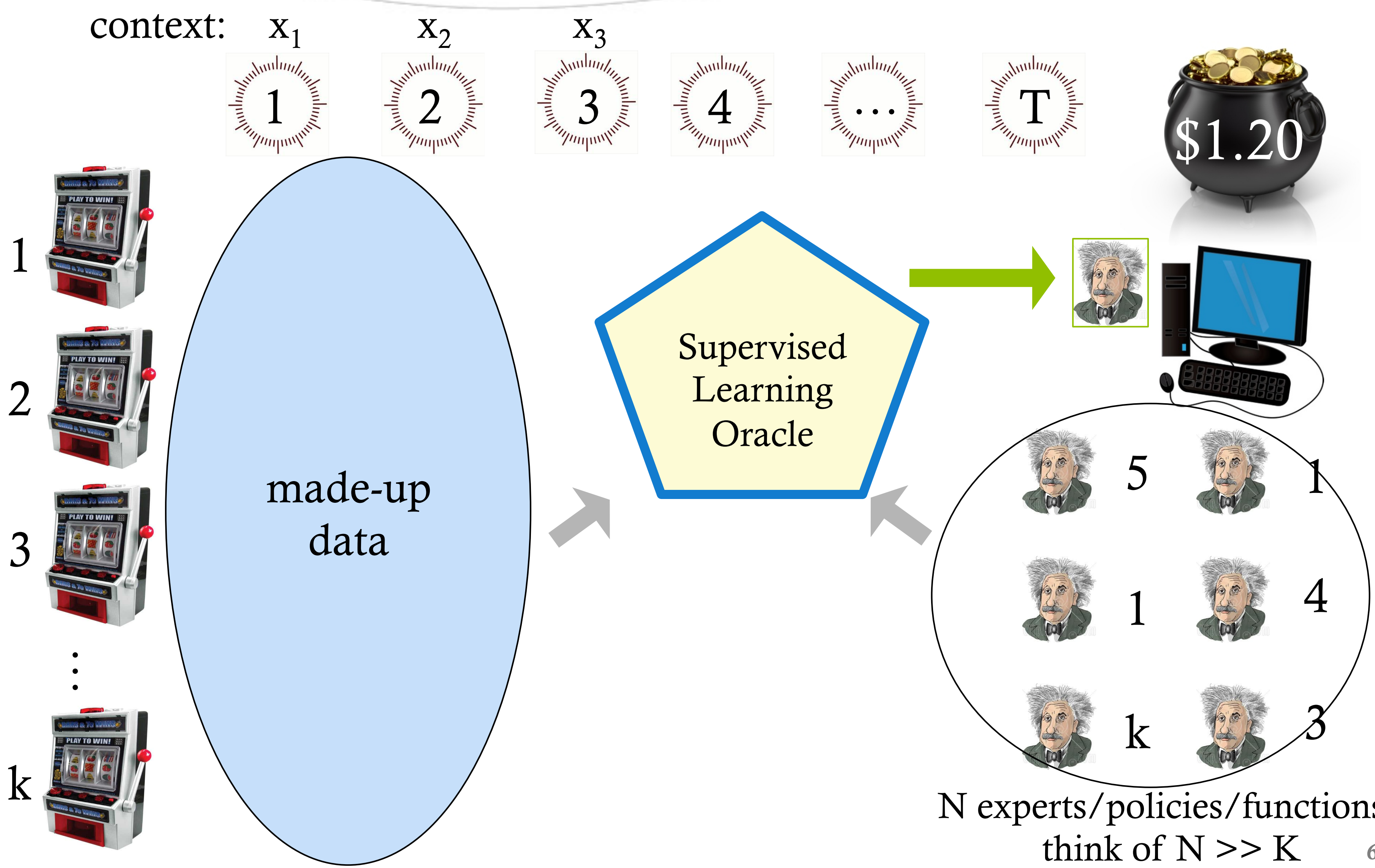
Back to Contextual Bandits

N experts/policies/functions
think of N >> K

Back to Contextual Bandits



Back to Contextual Bandits



Randomized-UCB

Main Theorem [Dudik-Hsu-Kale-Karampatziakis-Langford-**R**-Zhang '11]: For any $\delta > 0$, w.p. at least $1-\delta$, given access to a supervised learning oracle, Randomized-UCB has regret at most $O((KT \ln(NT/\delta))^{1/2} + K \ln(NK/\delta))$ in the stochastic contextual bandit setting and runs in time poly(K,T, ln N).

Randomized-UCB

Main Theorem [Dudik-Hsu-Kale-Karampatziakis-Langford-**R**-Zhang '11]:
For any $\delta > 0$, w.p. at least $1-\delta$, given access to a supervised learning oracle, Randomized-UCB has regret at most $O((KT \ln(NT/\delta))^{1/2} + K \ln(NK/\delta))$ in the stochastic contextual bandit setting and runs in time poly(K,T, ln N).

if arms are chosen among only good policies s.t. all have variance < approx 2K, we win
can prove this exists via a minimax theorem

↓

this condition can be softened to occasionally allow choosing of bad policies
via "randomized" upper confidence bounds

↓

creates a problem of how to choose arms as to satisfy the constraints
expressed as convex optimization problem

↓

solvable by ellipsoid algorithm
can implement a separation oracle with the supervised learning oracle

Randomized-UCB

Main Theorem [Dudik-Hsu-Kale-Karampatziakis-Langford-**R**-Zhang '11]:
For any $\delta > 0$, w.p. at least $1-\delta$, given access to a supervised learning oracle, Randomized-UCB has regret at most $O((KT \ln(NT/\delta))^{1/2} + K \ln(NK/\delta))$ in the stochastic contextual bandit setting and runs in time poly(K,T, ln N).

if arms are chosen among only good policies s.t. all have variance < approx 2K, we win

can prove this exists via a minimax theorem

this condition can be softened to occasionally allow choosing of bad policies

via "randomized" upper confidence bounds

creates a problem of how to choose arms as to satisfy the constraints

expressed as convex optimization problem

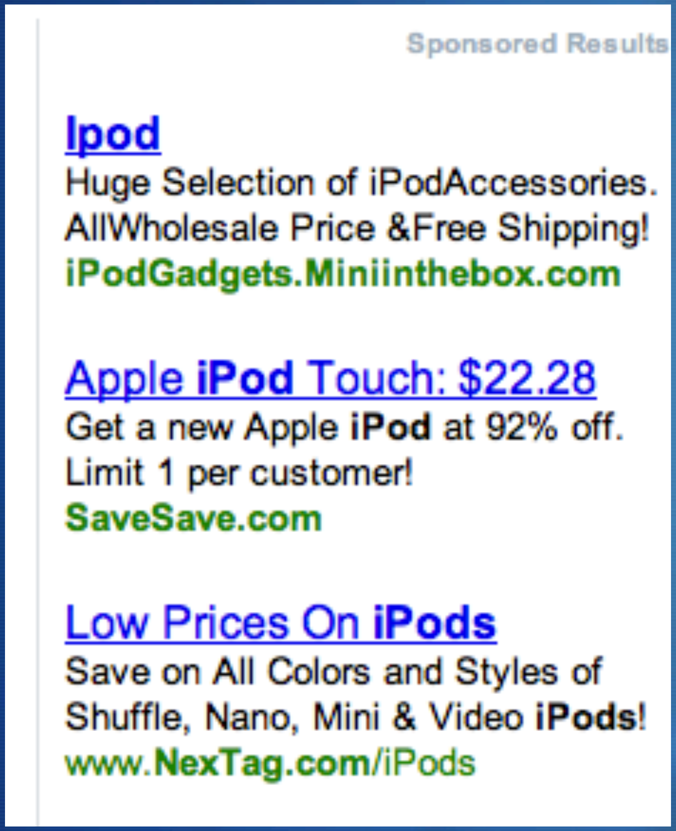
solvable by ellipsoid algorithm

can implement a separation oracle with the supervised learning oracle

Not practical to implement! (yet)

Outline

- The setting and some background
- Show ideas that fail
- Give a high probability optimal algorithm
- Dealing with VC sets
- An efficient algorithm
- **Slates**



Bandit Slate Problems

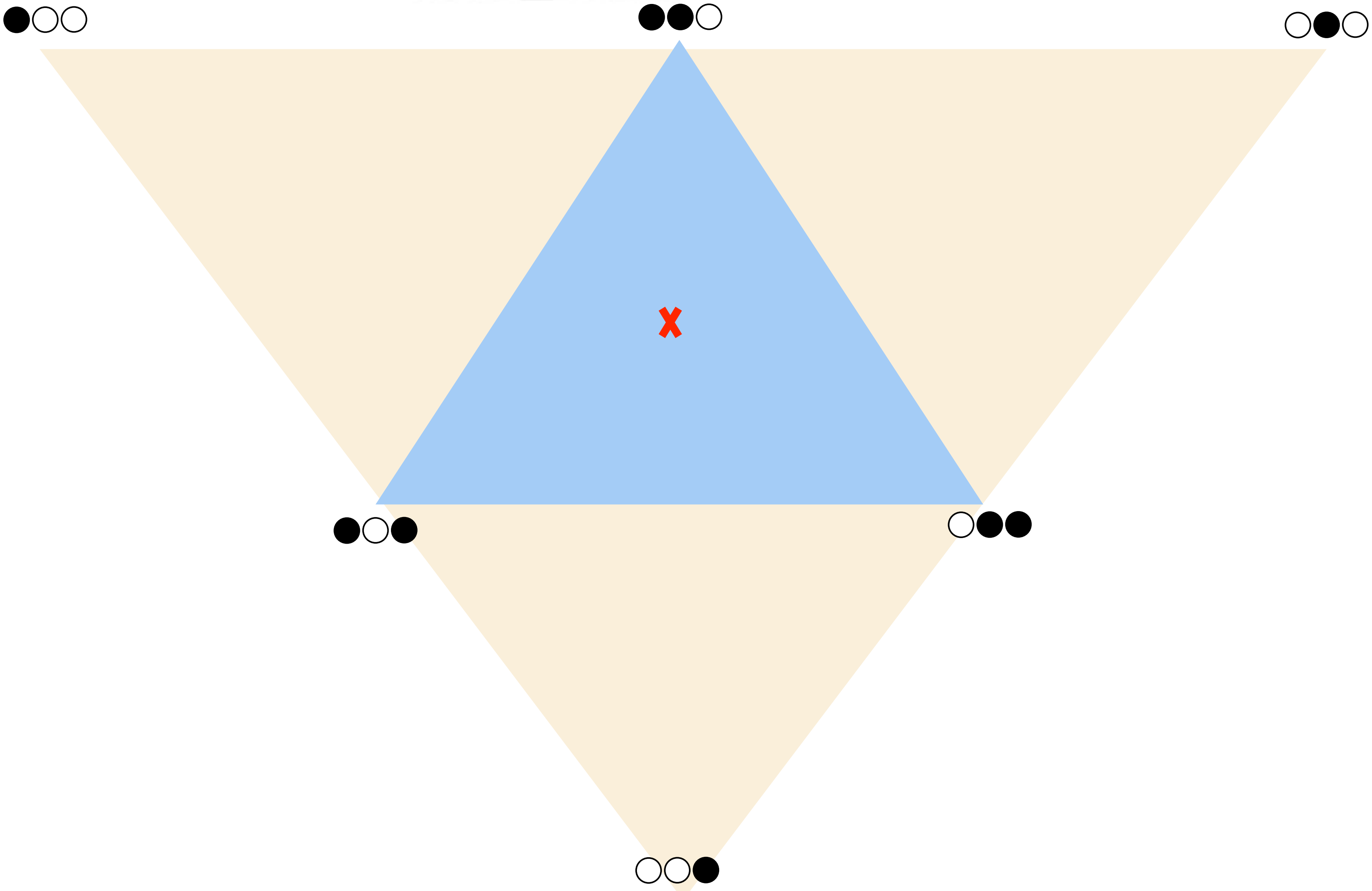
[Kale-**R**-Schapire '11]

Problem: Instead of selecting one arm, we need to select $s \geq 1$, arms (possibly ranked). The motivation is web ads where a search engine shows multiple ads at once.

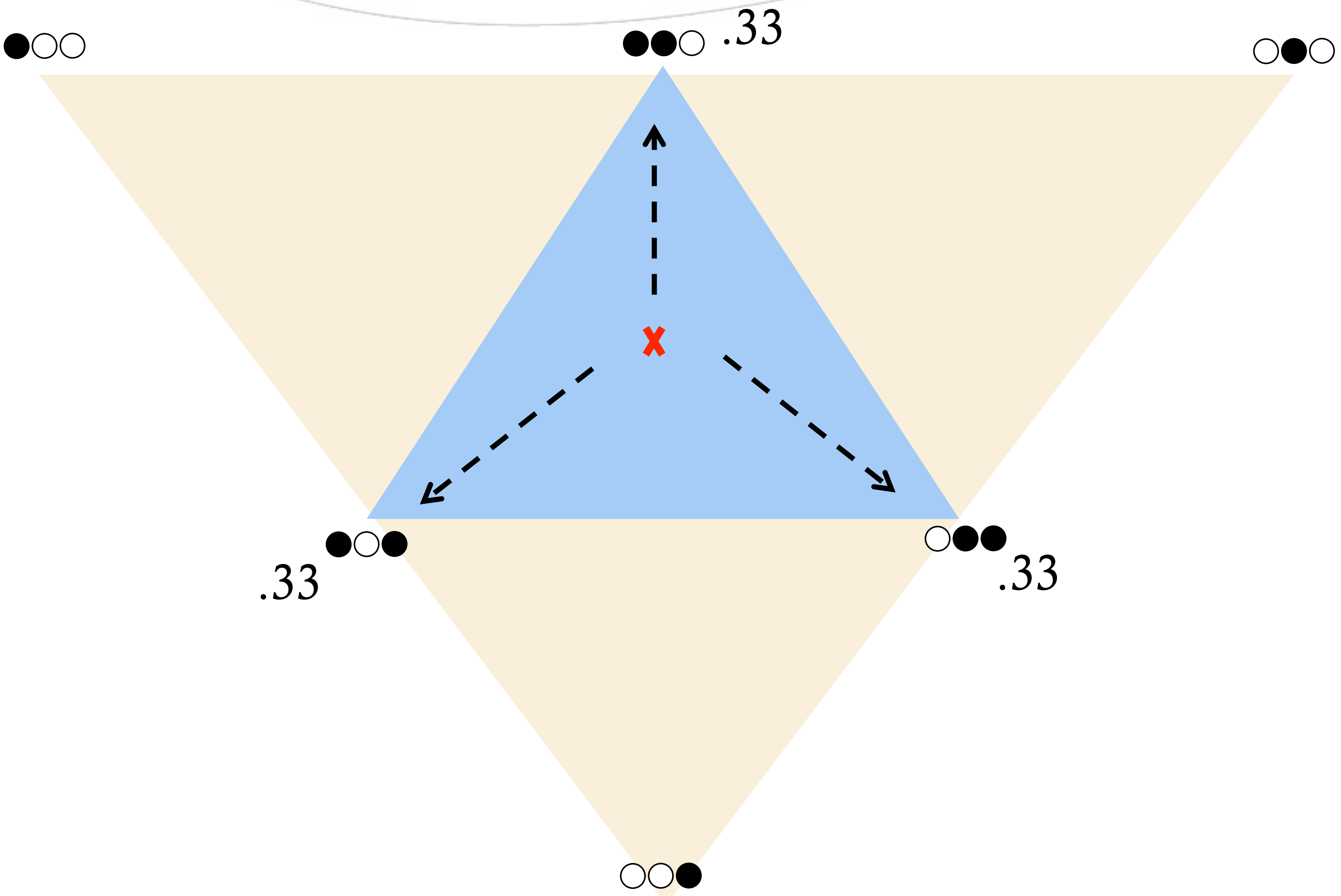
Slates Setting

- On round t algorithm selects a sate S_t of s arms
  - **Unordered** or Ordered
  - **No context** or Contextual
- Algorithm sees $r_j(t)$ for all j in S.
- Algorithm gets reward $\sum_{j \in S} r_j(t)$
- Obvious solution is to reduce to the regular bandit problem, but we can do much better.

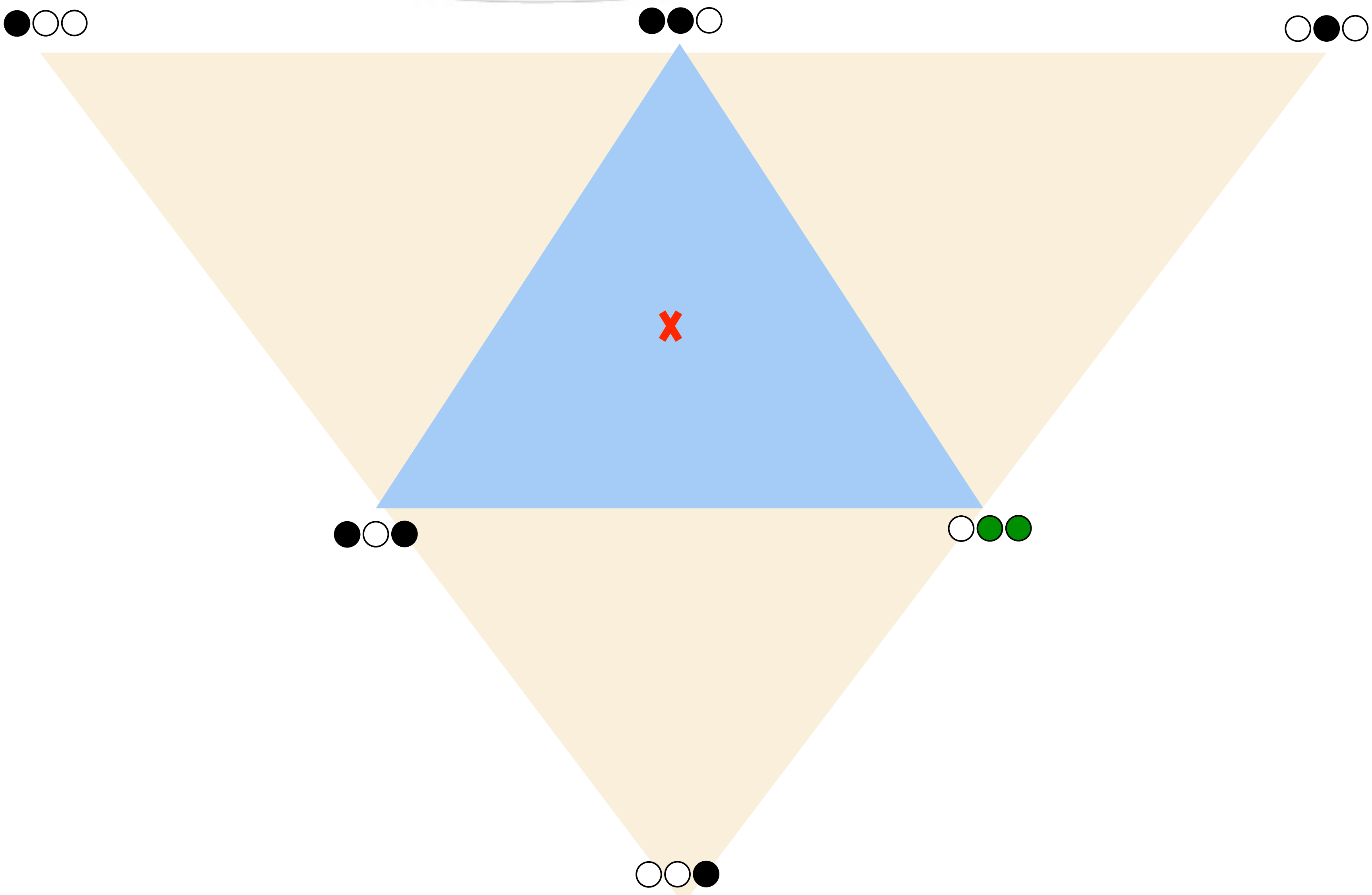
Algorithm Idea



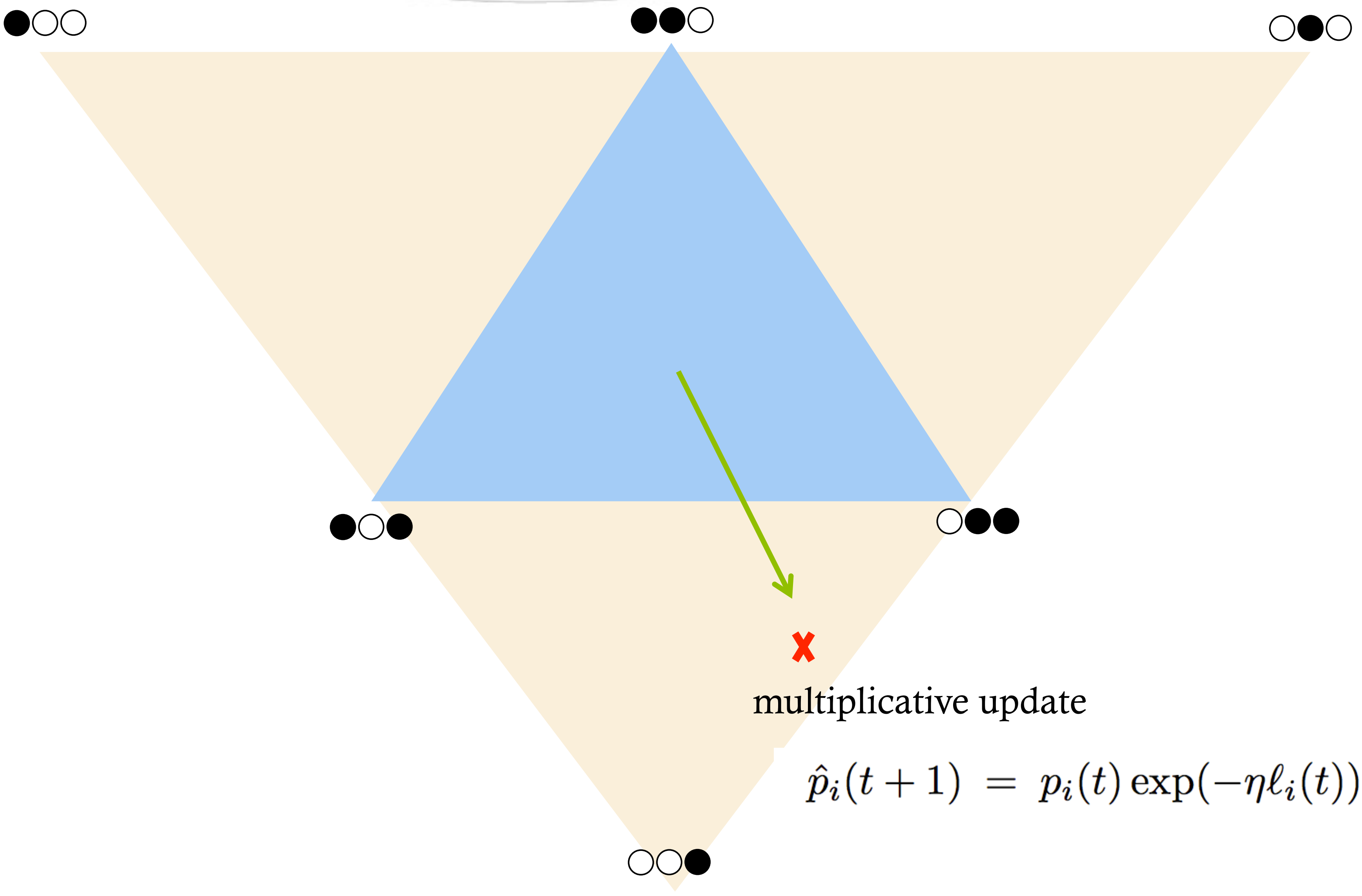
Algorithm Idea



Algorithm Idea

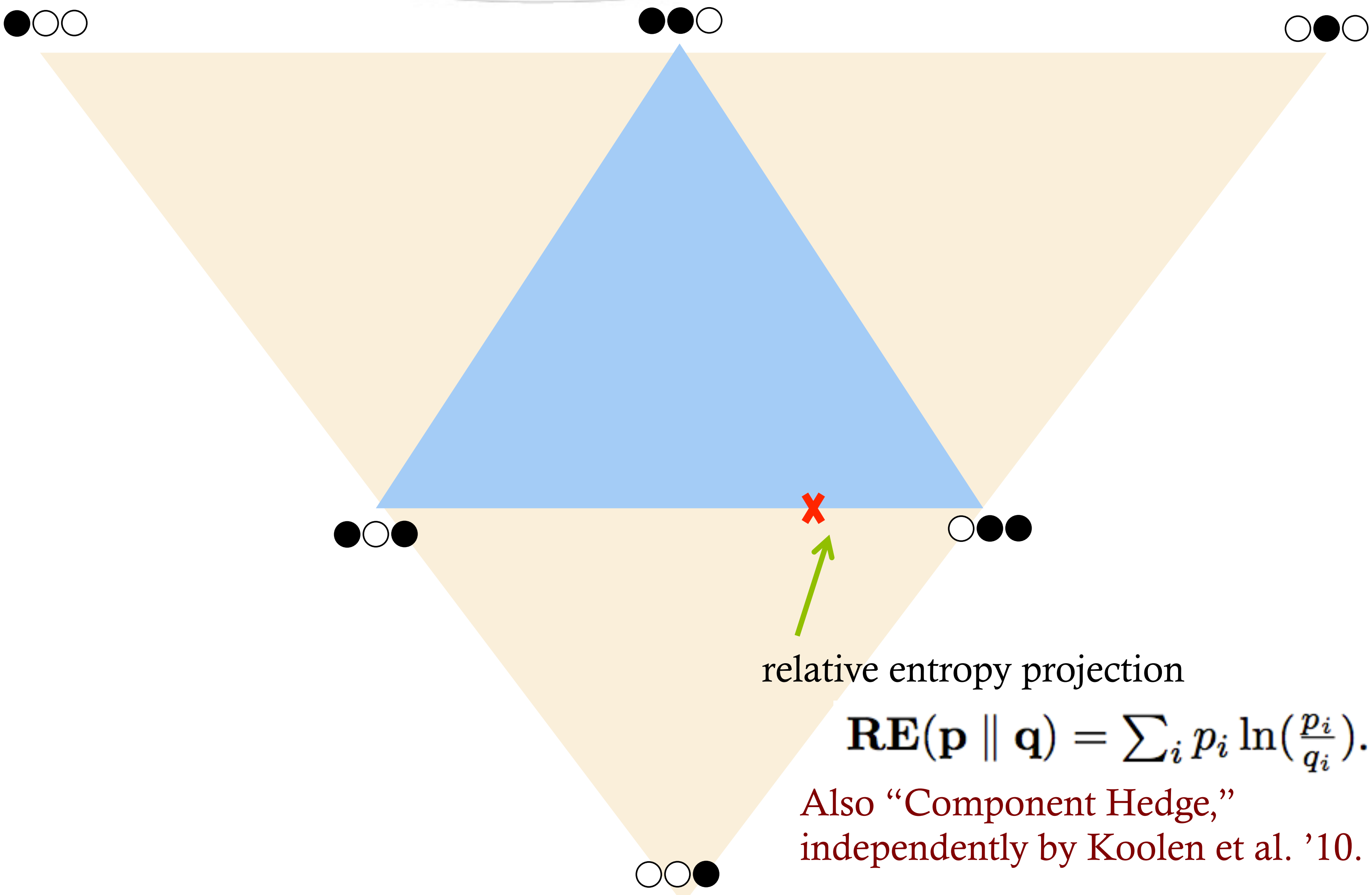


Algorithm Idea



$$\hat{p}_i(t+1) \;=\; p_i(t)\exp(-\eta\ell_i(t))/Z(t)$$

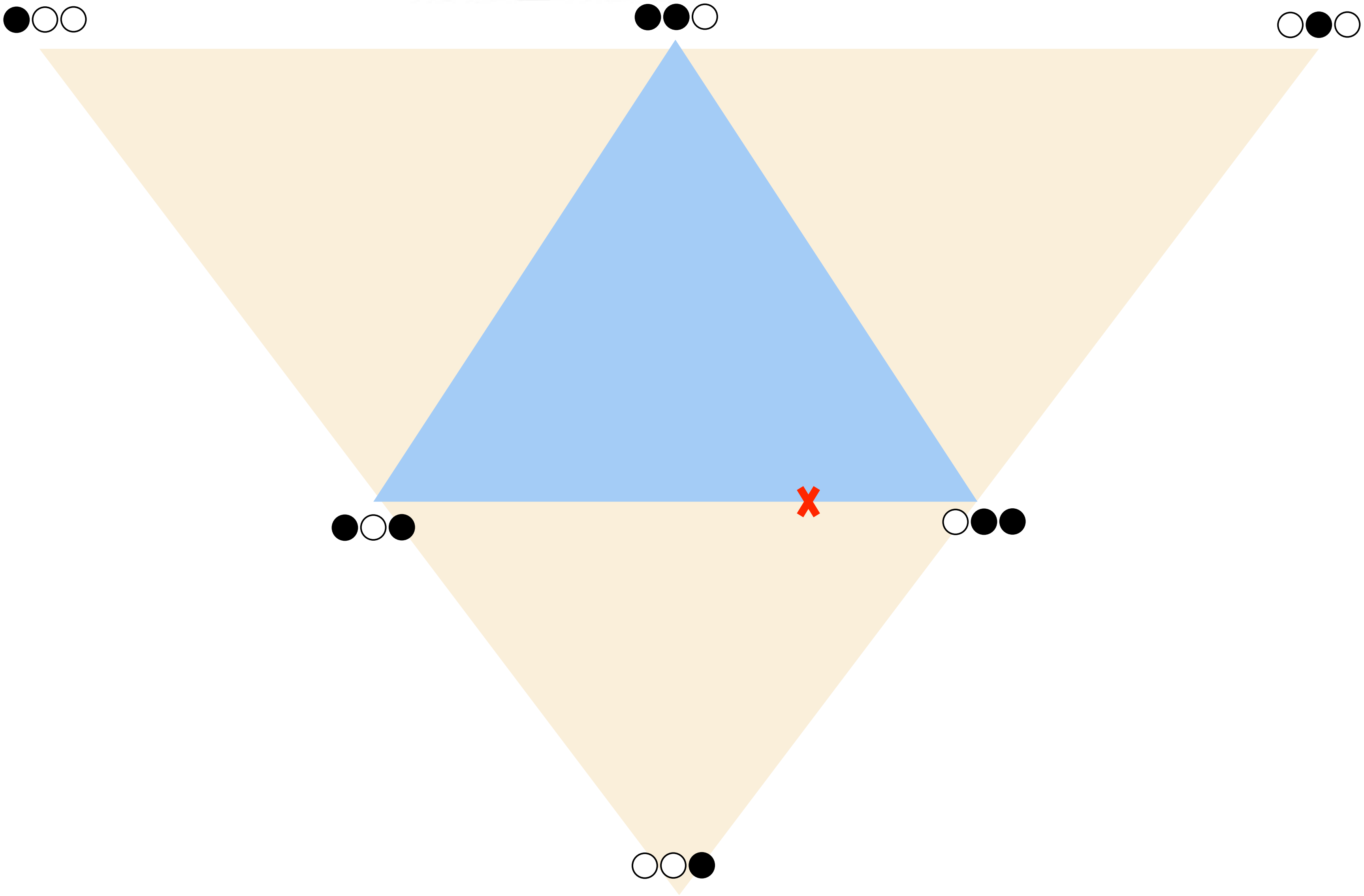
Algorithm Idea



$$\mathbf{RE}(\mathbf{p} \parallel \mathbf{q}) = \sum_i p_i \ln(\frac{p_i}{q_i}).$$

Also “Component Hedge,”
independently by Koolen et al. ’10.

Algorithm Idea



Slate Results

	Unordered Slates	Ordered, with Positional Factors
No Policies	$\tilde{O}(sKT)^{1/2}$ *	$\tilde{O}(s(KT)^{1/2})$
N Policies	$\tilde{O}(sKT \ln N)^{1/2}$	$\tilde{O}(s(KT \ln N)^{1/2})$

*Independently obtained by Uchiya et al. '10, using different methods.

Discussion

- The contextual bandit setting captures many interesting real-world problems.
- We presented the first optimal, high-probability, contextual algorithm.
- We showed how one could possibly make it efficient.
  - Not fully there yet…
- We discussed slates – a more real-world setting.
  - How to make those efficient?