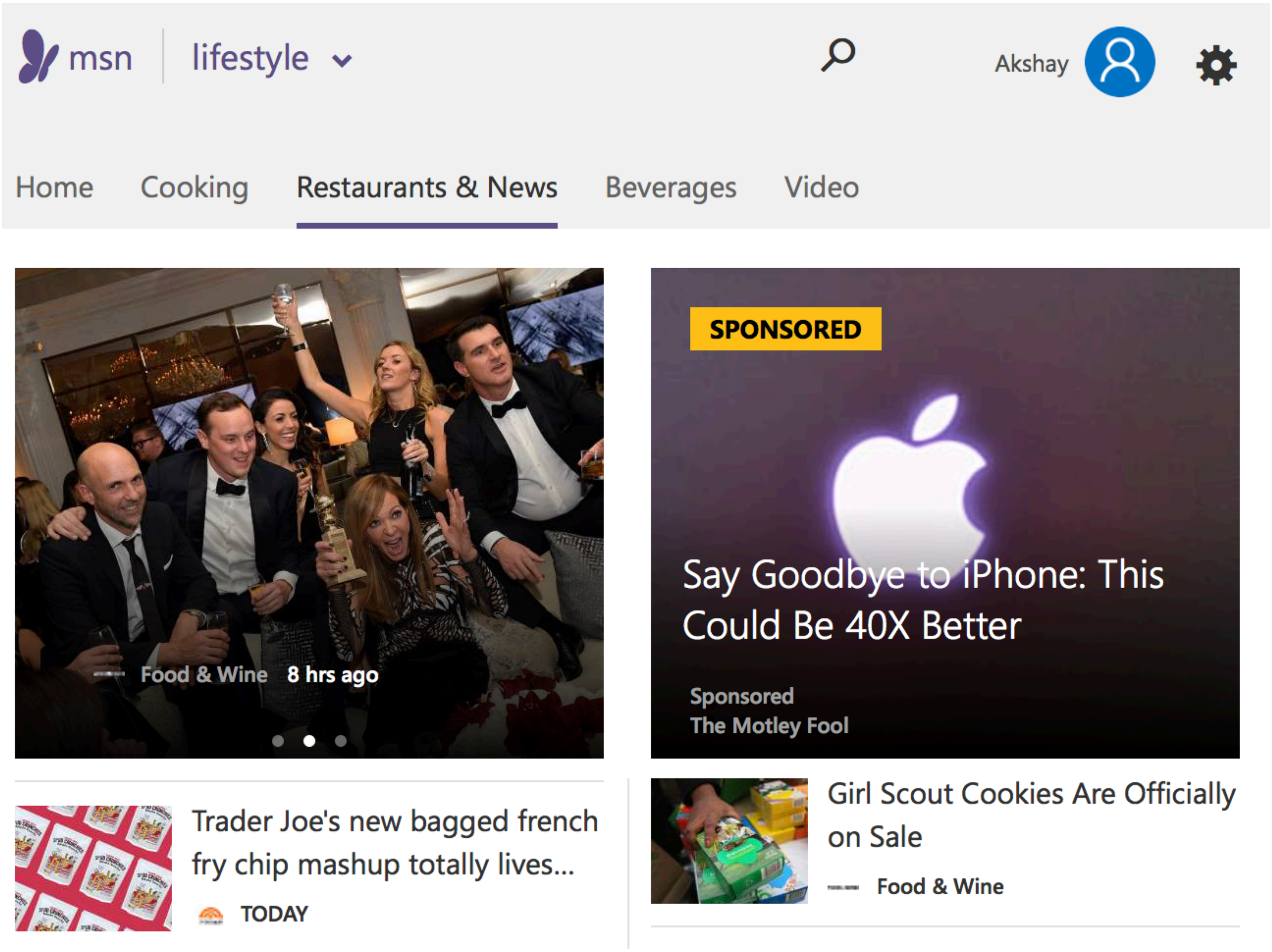
Online and Offline Experimentation in Complex Systems

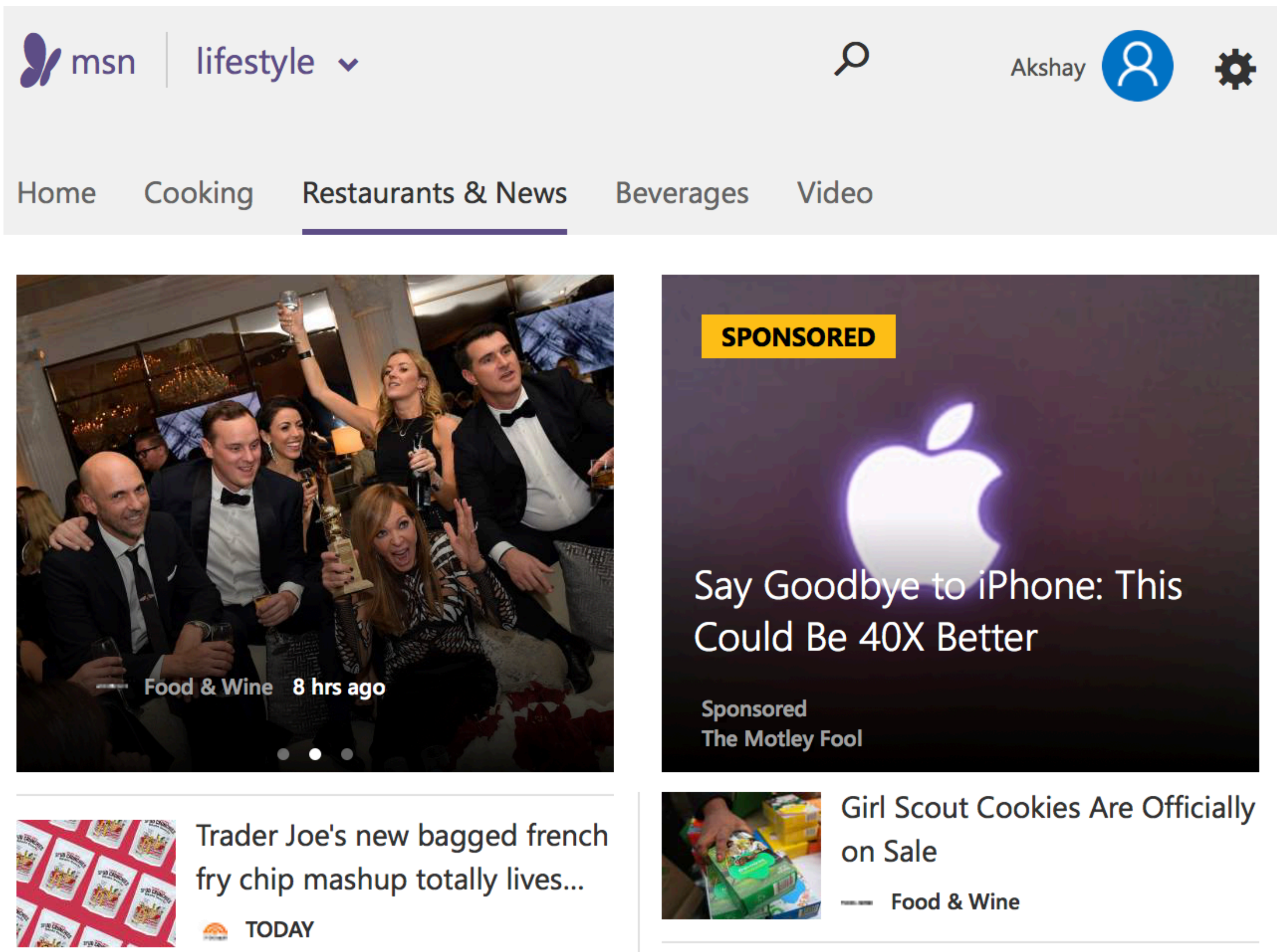
Akshay Krishnamurthy
Microsoft Research, NYC
akshay@cs.umass.edu

Online Personalization



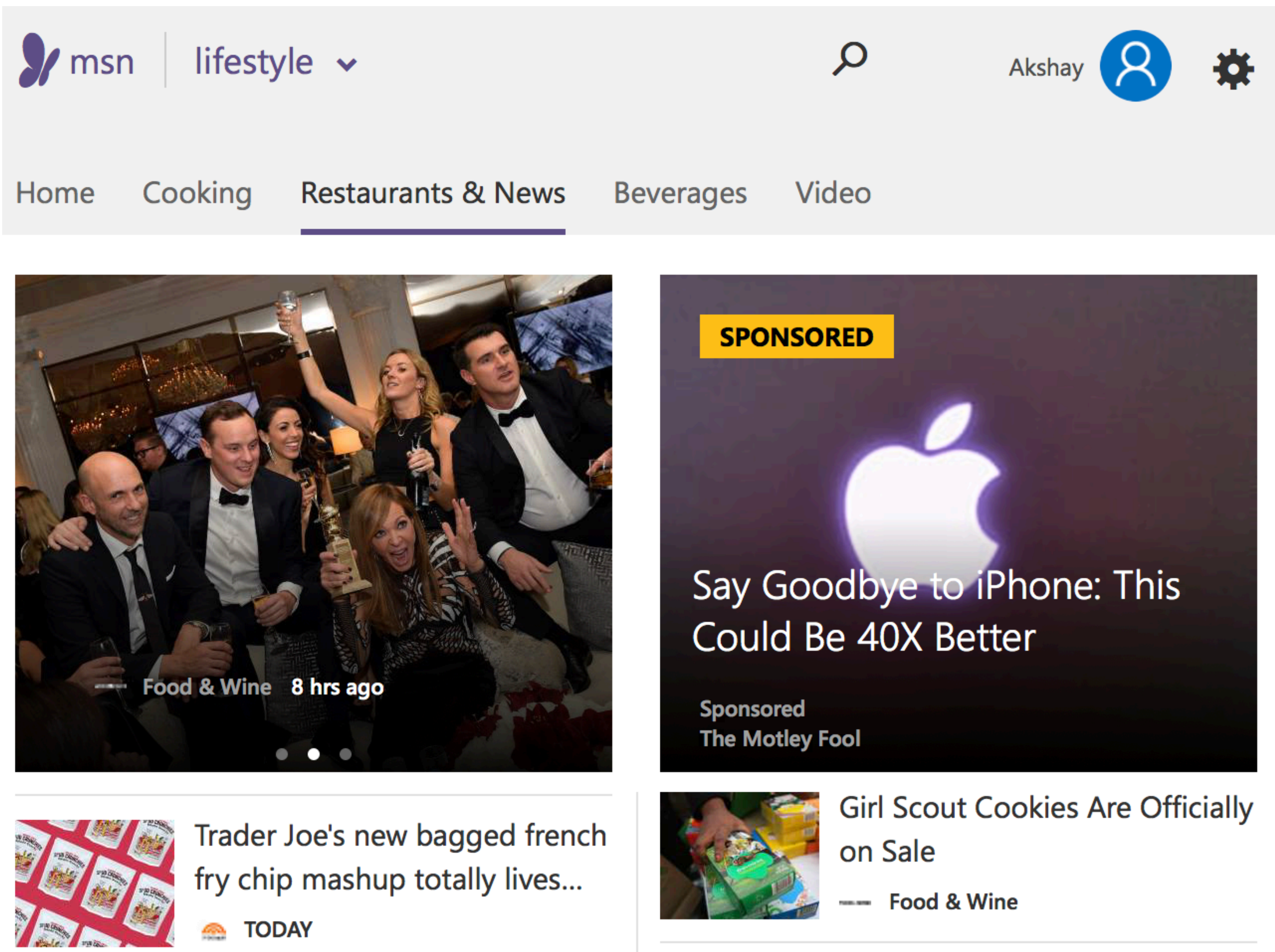
- Learn from interacting with users in production

Online Personalization



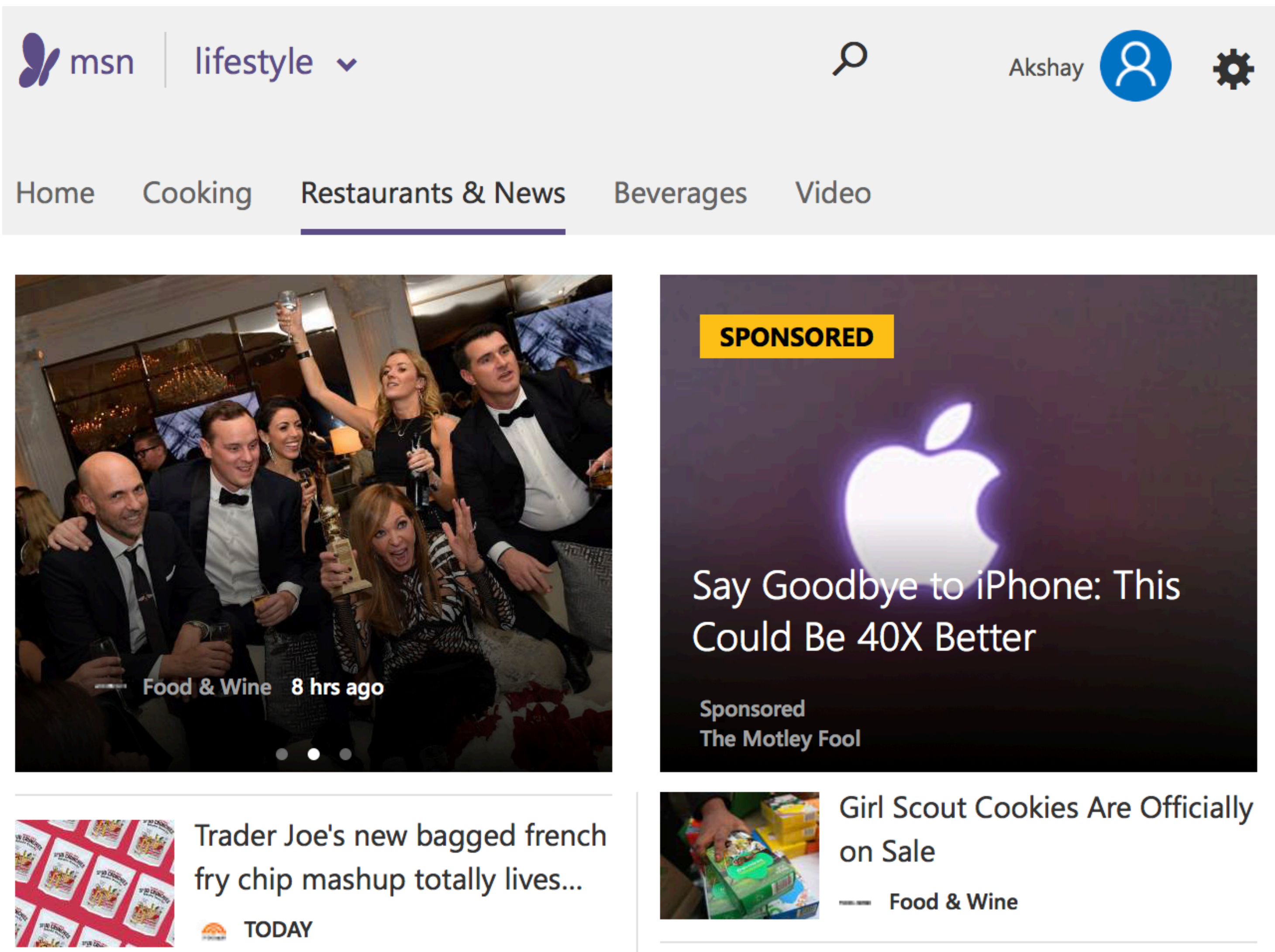
- Learn from interacting with users in production
- No counterfactuals

Online Personalization



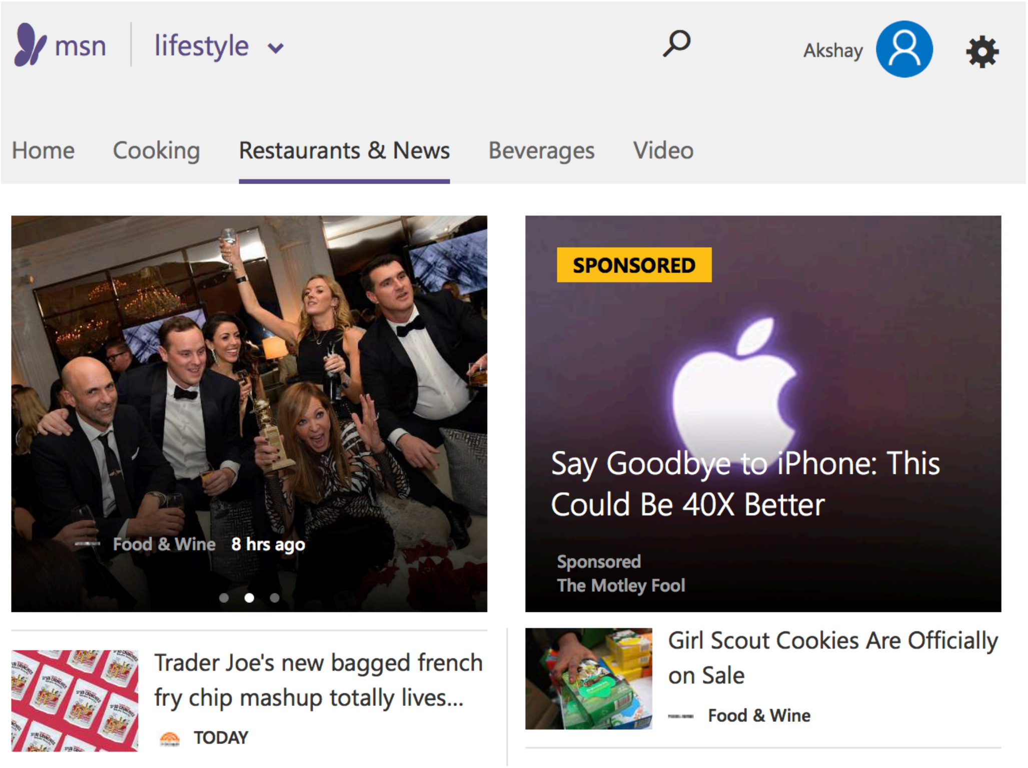
- Learn from interacting with users in production
- No counterfactuals
- Exploration vs Exploitation

Online Personalization



- Learn from interacting with users in production
- No counterfactuals
- Exploration vs Exploitation
- Optimize whole-page layout

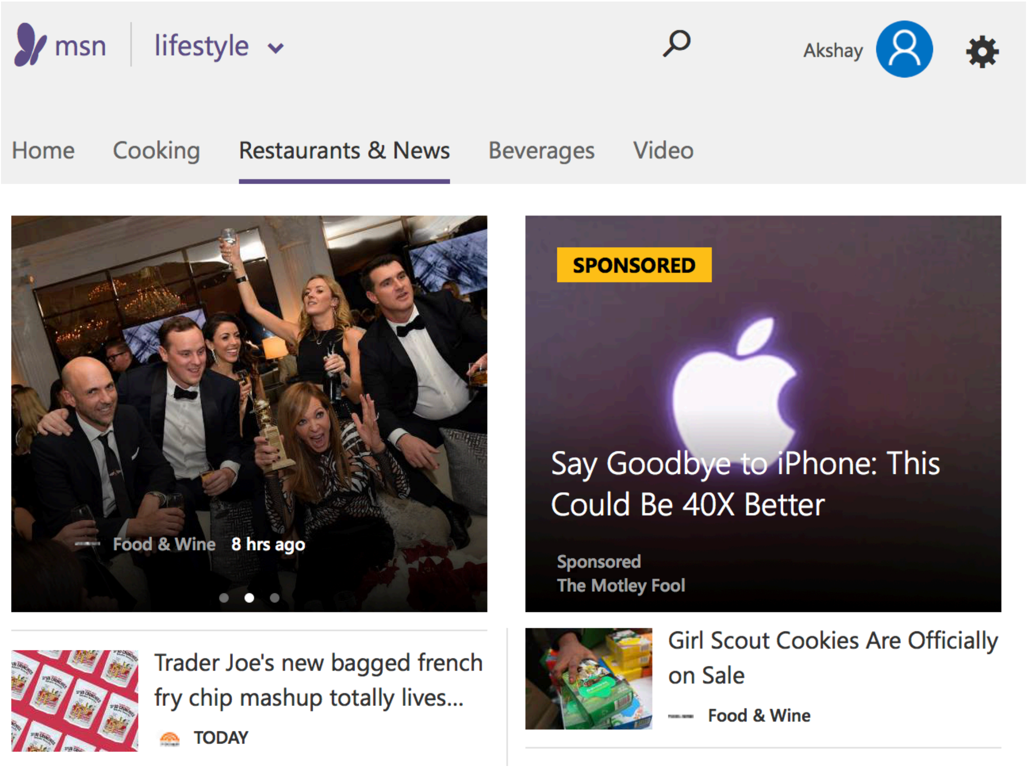
Industry Standard: A/B Testing



Industry Standard: A/B Testing

Given policy π:

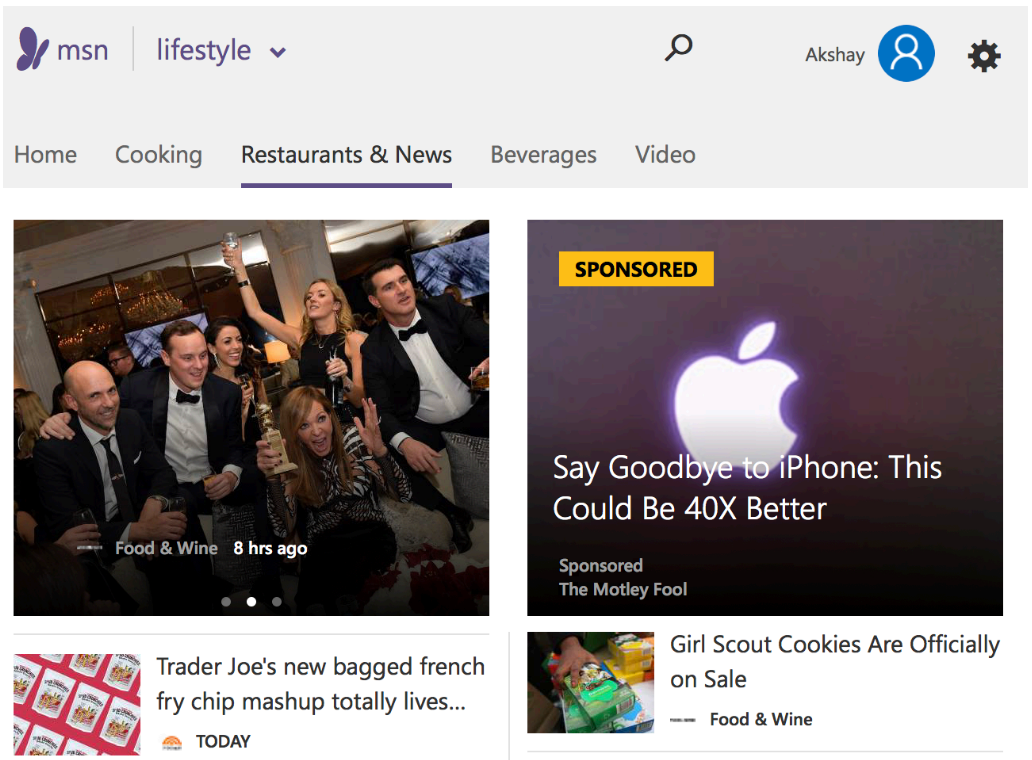
1. Use π for 1/2 of traffic (at random)
2. Evaluate π's quality (click prob.)



Industry Standard: A/B Testing

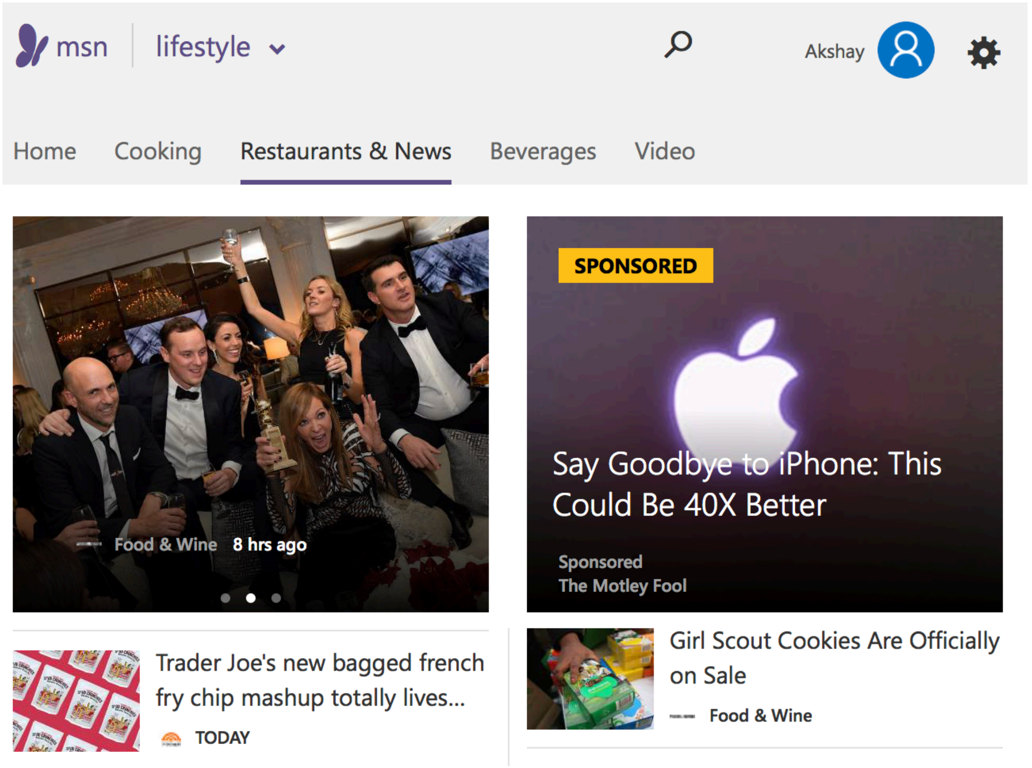
Given policy π:

1. Use π for 1/2 of traffic (at random)
2. Evaluate π's quality (click prob.)



Two main issues:

Industry Standard: A/B Testing



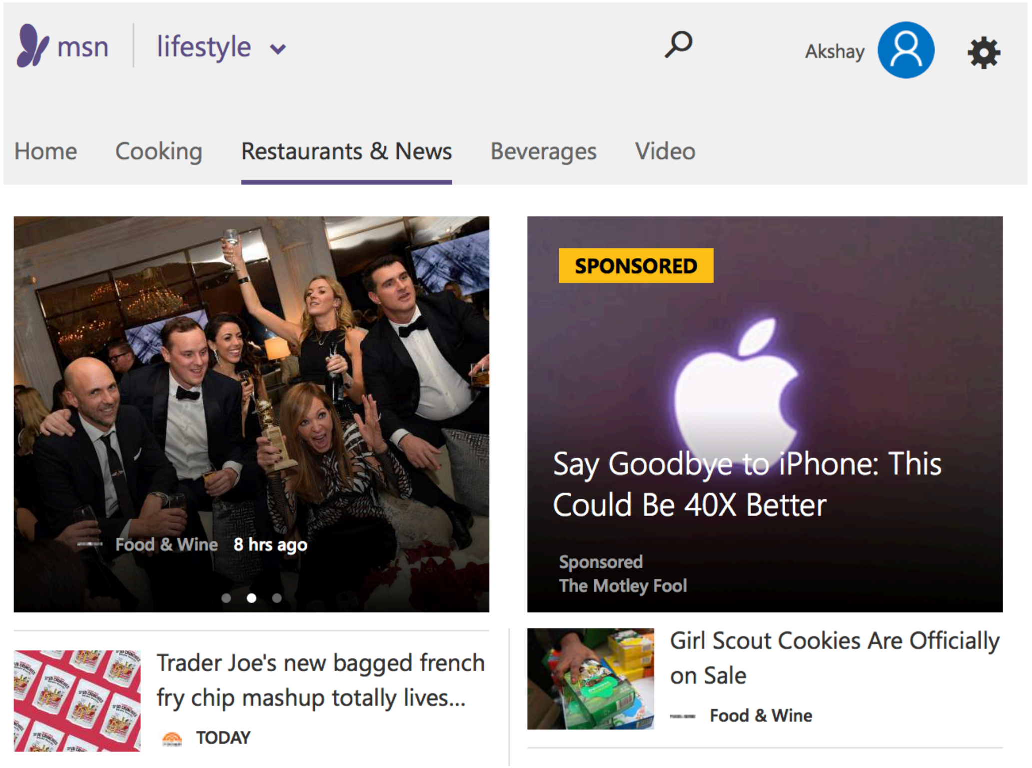
Given policy π:

1. Use π for 1/2 of traffic (at random)
2. Evaluate π's quality (click prob.)

Two main issues:

1. Poor performance while evaluating policies

Industry Standard: A/B Testing



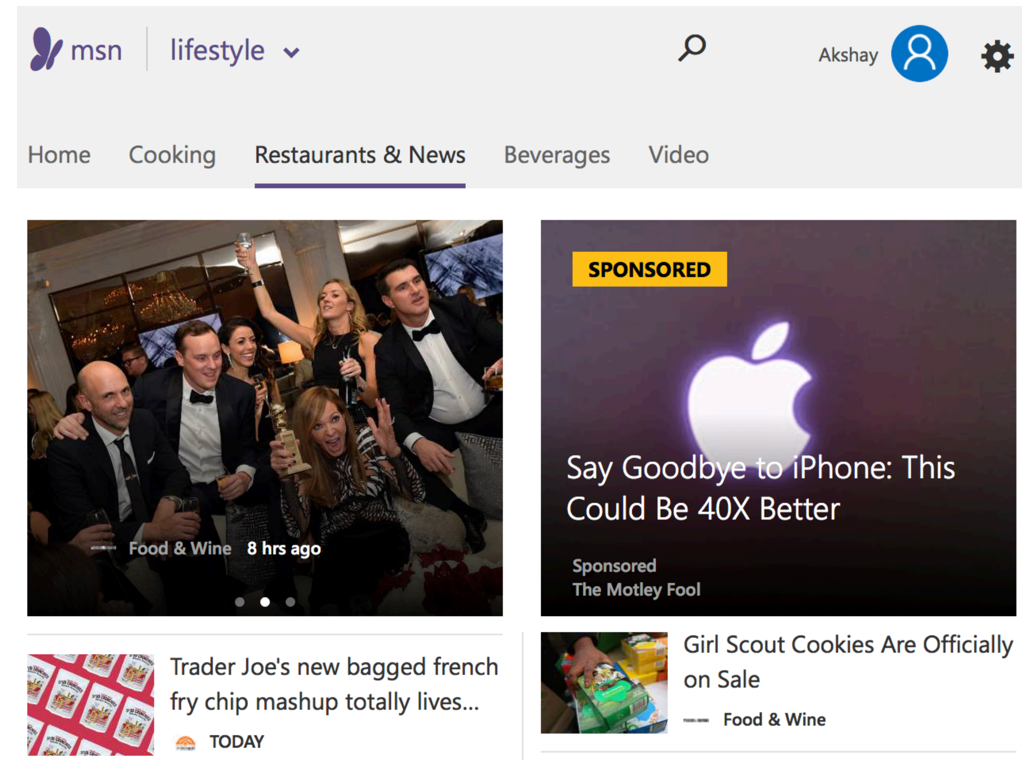
Given policy π:

1. Use π for 1/2 of traffic (at random)
2. Evaluate π's quality (click prob.)

Two main issues:

1. Poor performance while evaluating policies
2. Requires $O(|\Pi|)$ samples to evaluate $|\Pi|$ policies

Industry Standard: A/B Testing



Given policy π:

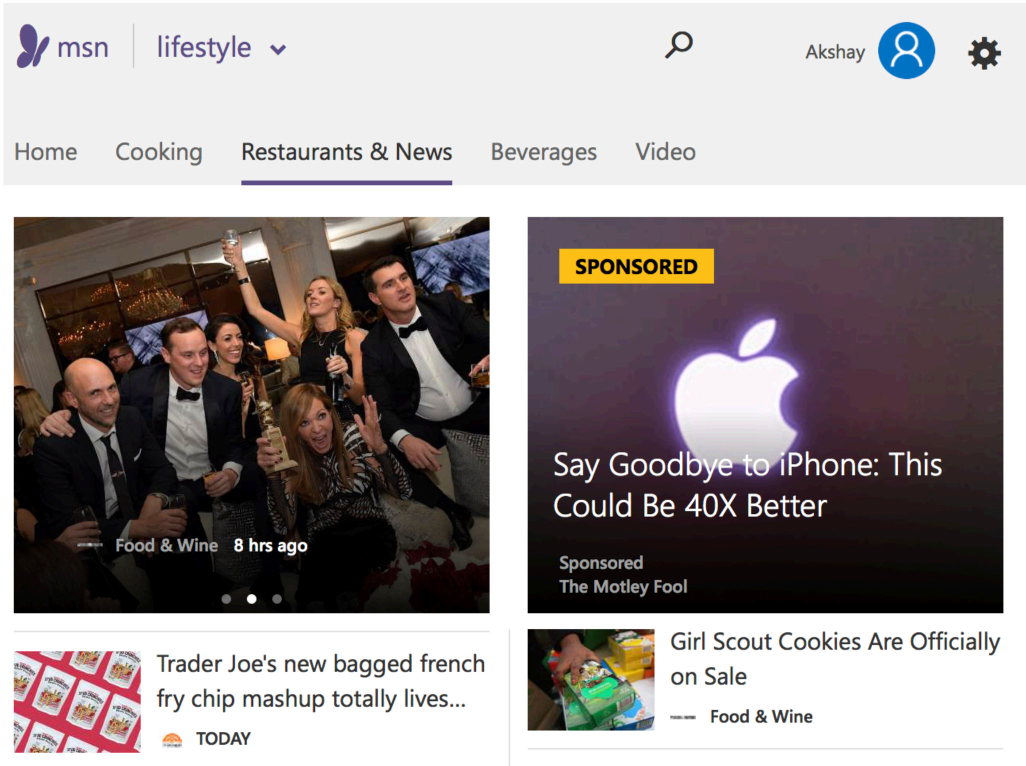
1. Use π for 1/2 of traffic (at random)
2. Evaluate π's quality (click prob.)

Two main issues:

1. Poor performance while evaluating policies
2. Requires $O(|\Pi|)$ samples to evaluate $|\Pi|$ policies

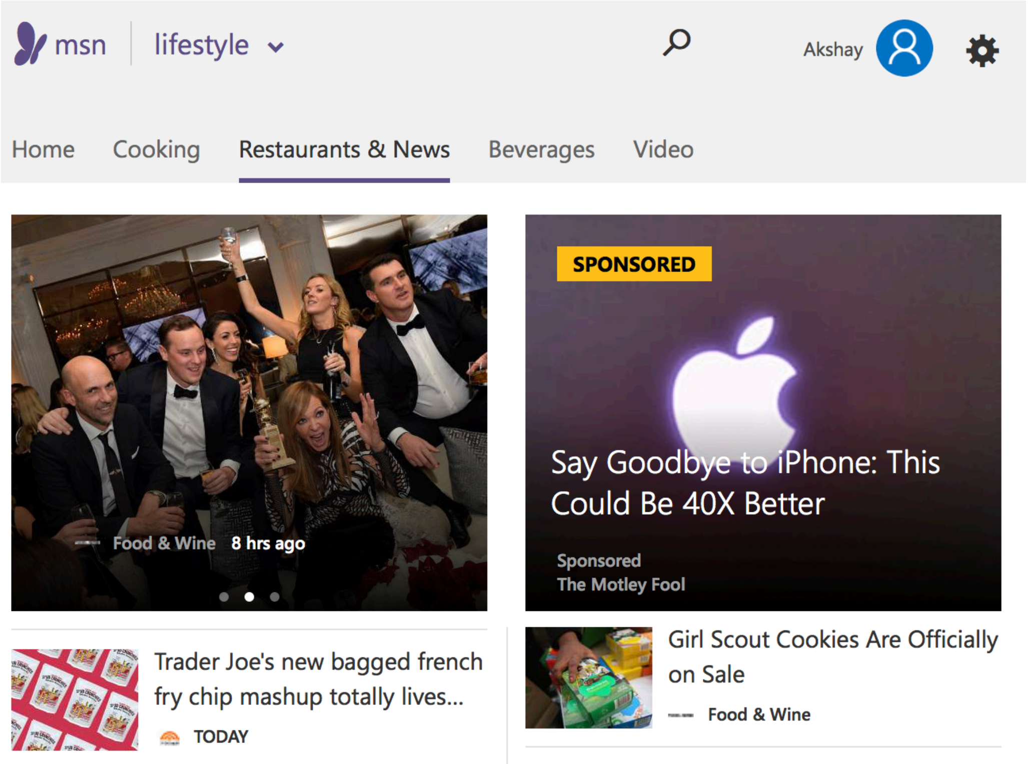
Can do **exponentially** better with contextual bandits!

Exploration + Offline Evaluation



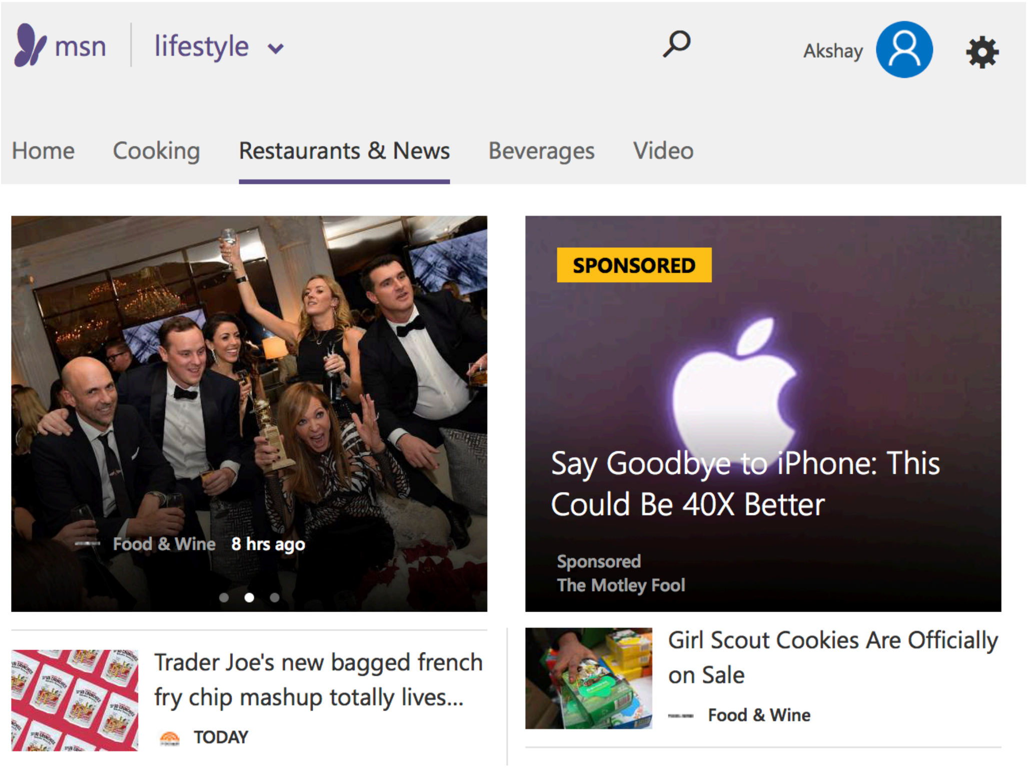
Exploration + Offline Evaluation

1. Collect dataset by serving content at random



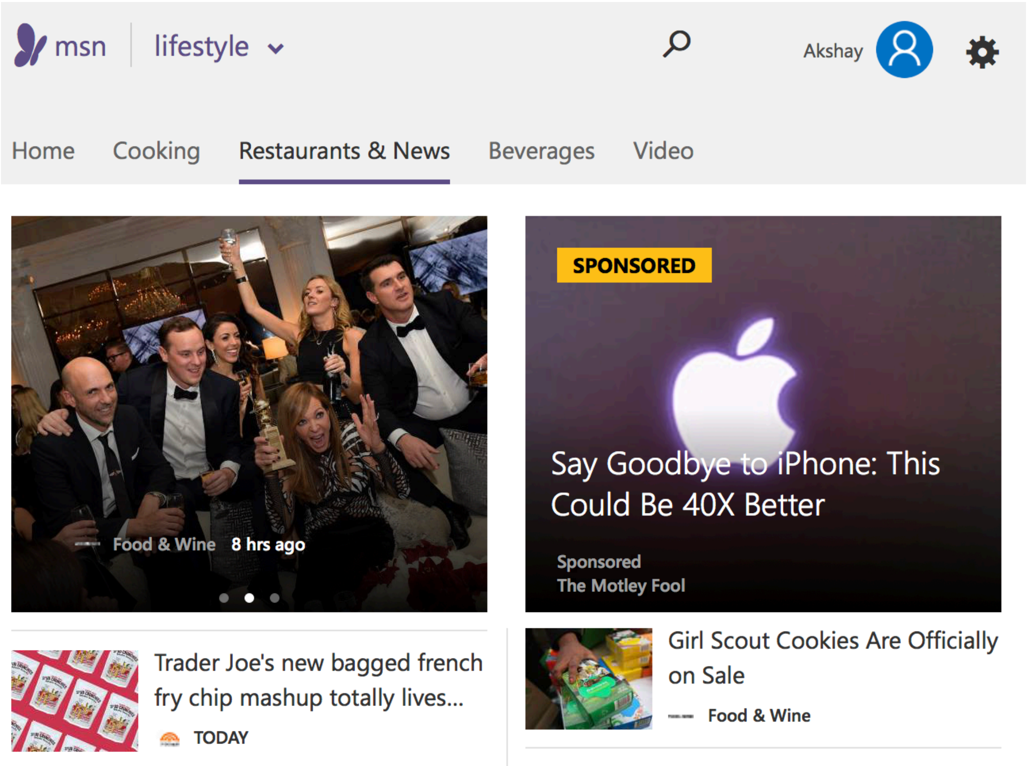
Exploration + Offline Evaluation

1. Collect dataset by serving content at random
2. For each policy, estimate performance by taking samples where we used its recommendation



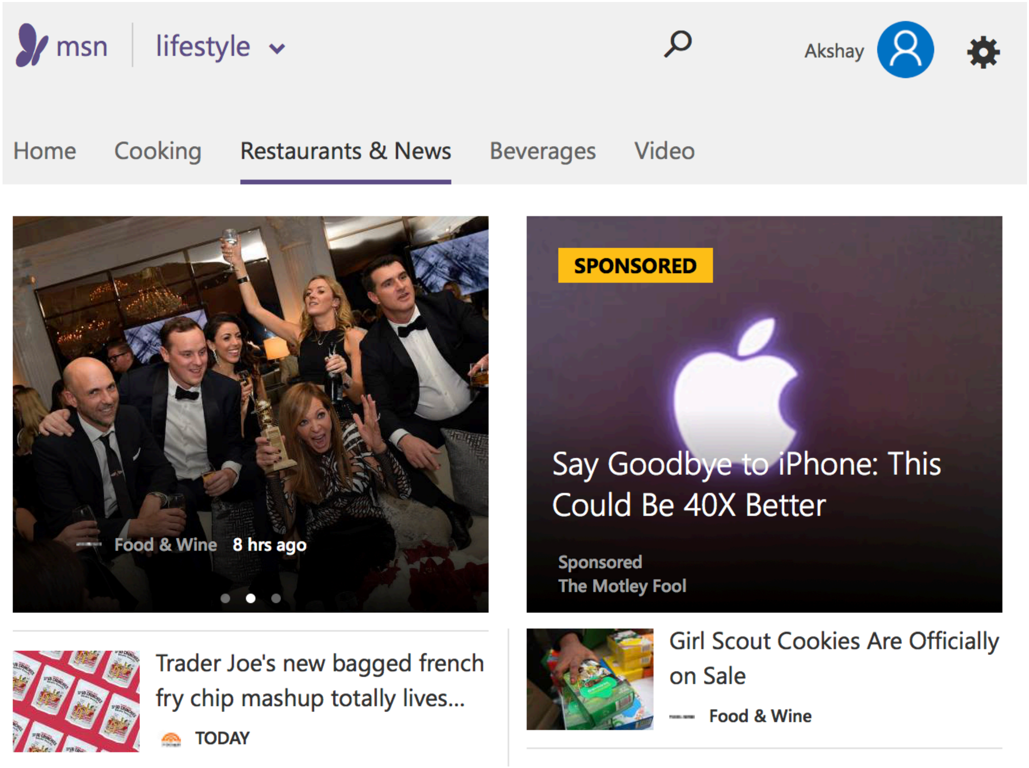
Exploration + Offline Evaluation

1. Collect dataset by serving content at random
2. For each policy, estimate performance by taking samples where we used its recommendation



With K actions and $|\Pi|$ policies, we need $O(K \log |\Pi|)$ samples

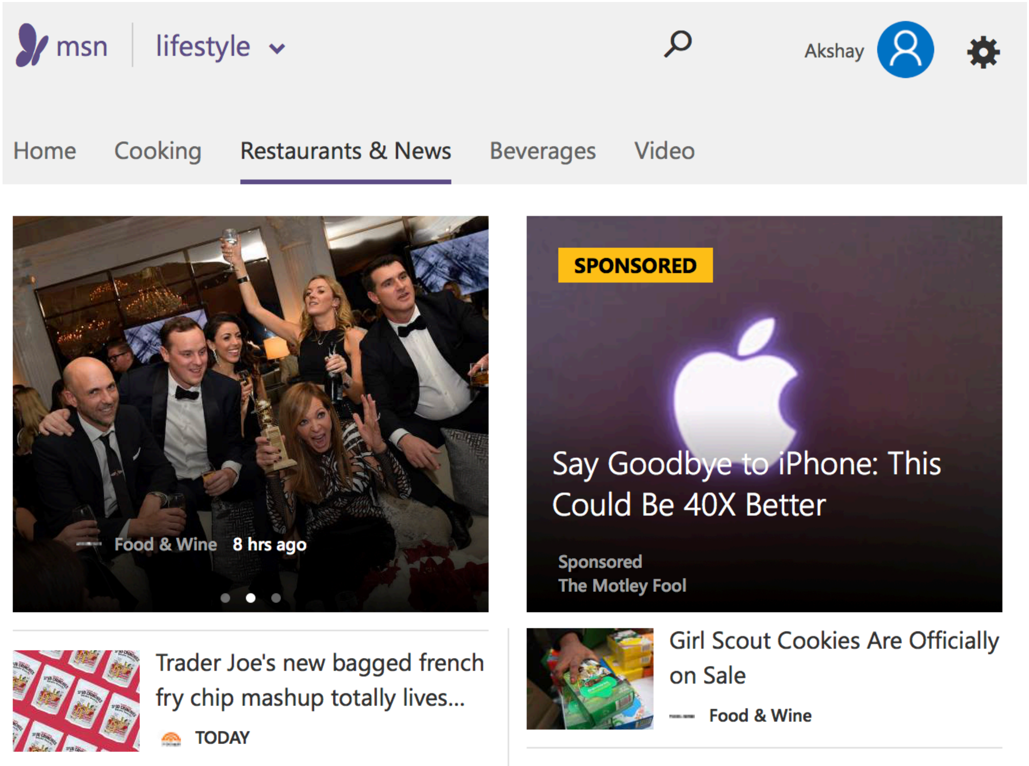
Contextual Bandits



Contextual Bandits

On each of T rounds:

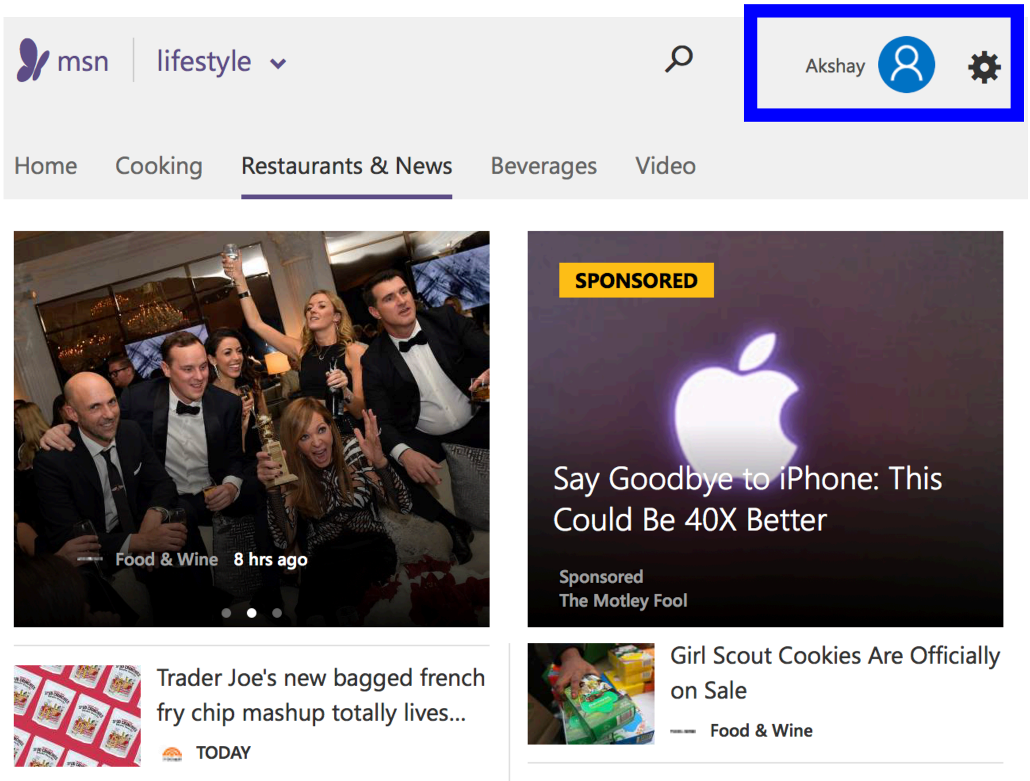
1. Observe context
2. Play action
3. Observe reward



Contextual Bandits

On each of T rounds:

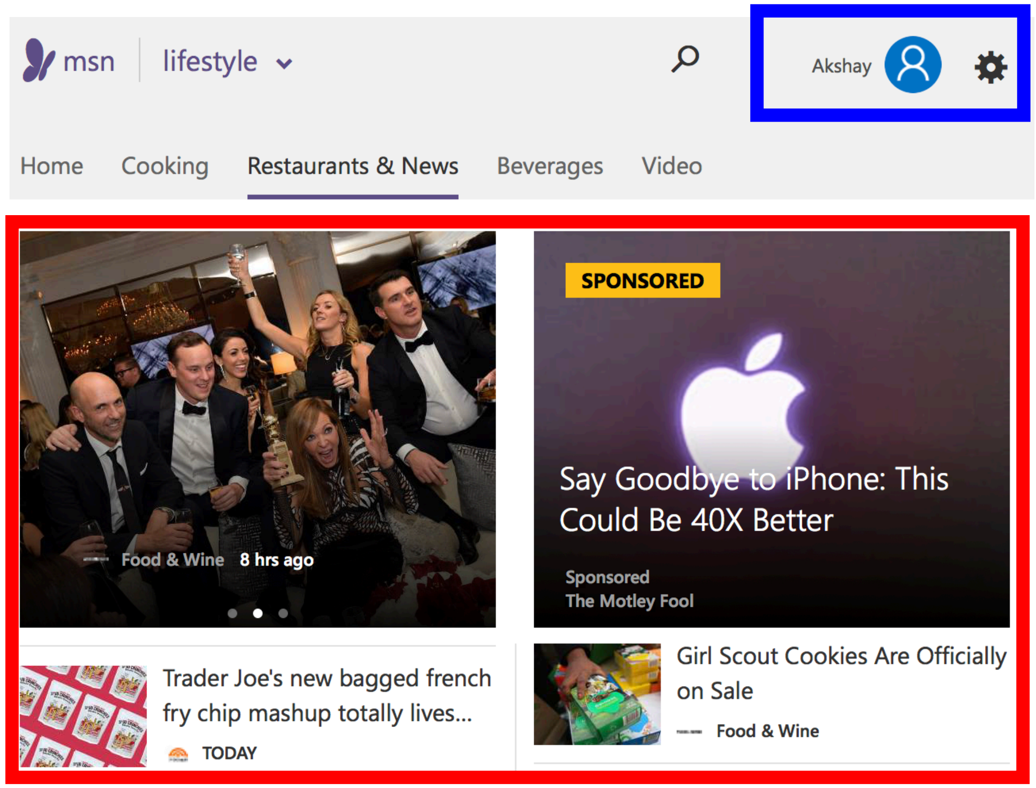
1. Observe context x_t
2. Play action
3. Observe reward



Contextual Bandits

On each of T rounds:

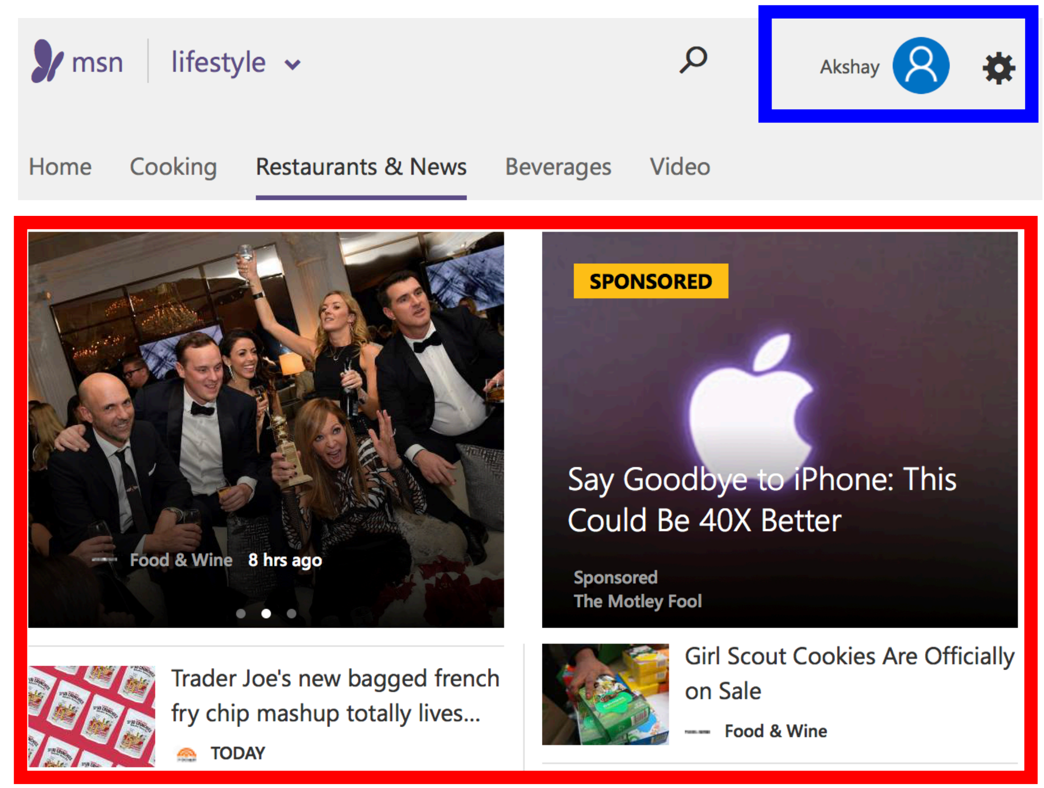
1. Observe context x_t
2. Play action a_t
3. Observe reward



Contextual Bandits

On each of T rounds:

1. Observe context x_t
2. Play action a_t
3. Observe reward $r_t(a_t, x_t)$



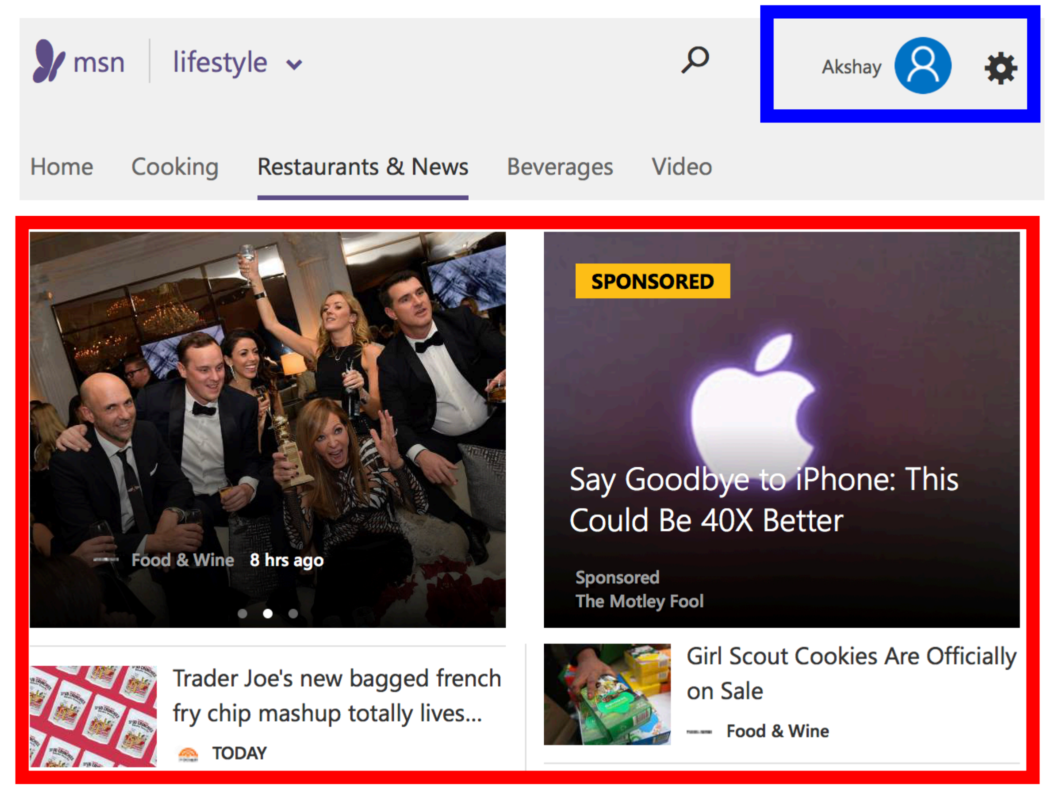
$r_t = \#\text{ clicks}$

Contextual Bandits

On each of T rounds:

1. Observe context x_t
2. Play action a_t
3. Observe reward $r_t(a_t, x_t)$

K = number of actions



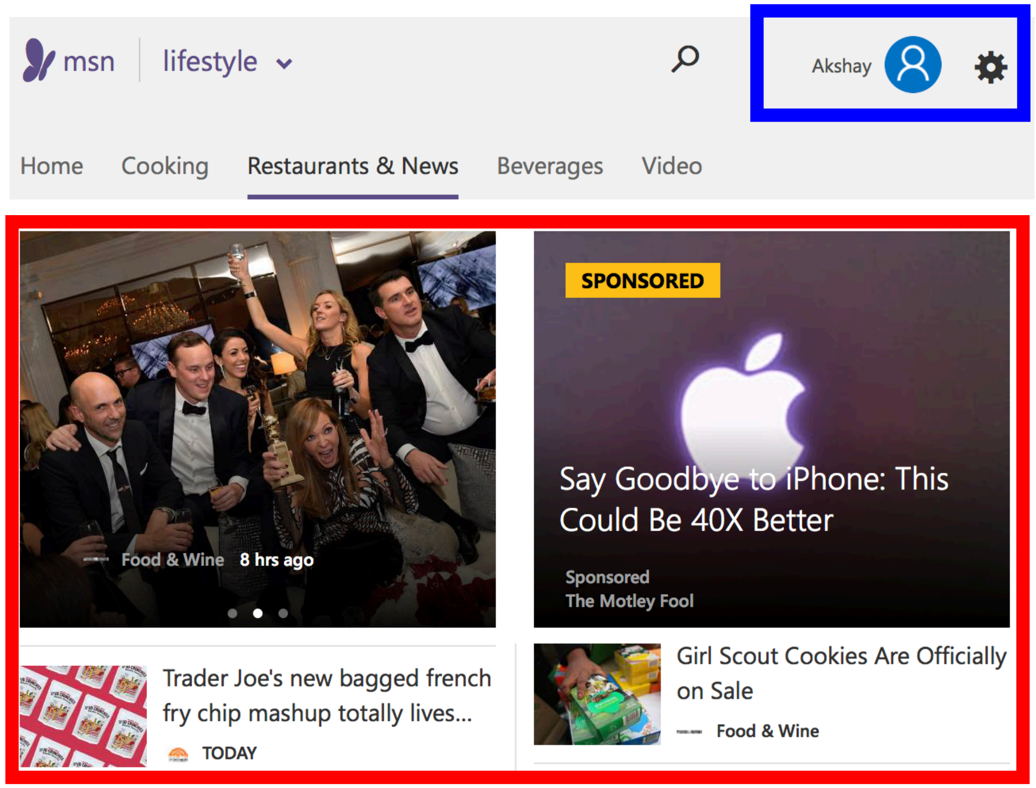
$r_t = \#\ \text{clicks}$

Contextual Bandits

On each of T rounds:

1. Observe context x_t
2. Play action a_t
3. Observe reward $r_t(a_t, x_t)$

K = number of actions



$$r_t = \#\ \text{clicks}$$

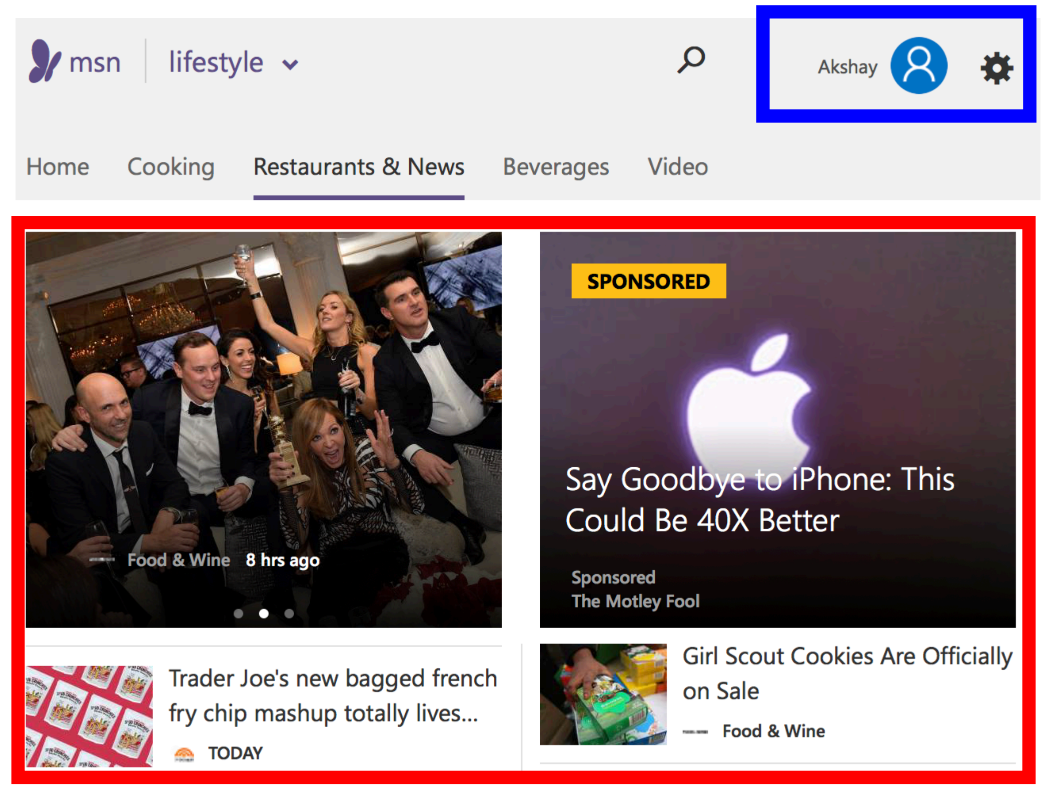
$$\text{Regret}(T, \Pi) = \max_{\pi \in \Pi} \text{Reward}(T, \pi) - \text{LearnerReward}(T)$$

Contextual Bandits

On each of T rounds:

1. Observe context x_t
2. Play action a_t
3. Observe reward $r_t(a_t, x_t)$

K = number of actions



$r_t = \#\ \text{clicks}$

$$\text{Regret}(T, \Pi) = \max_{\pi \in \Pi} \text{Reward}(T, \pi) - \text{LearnerReward}(T)$$

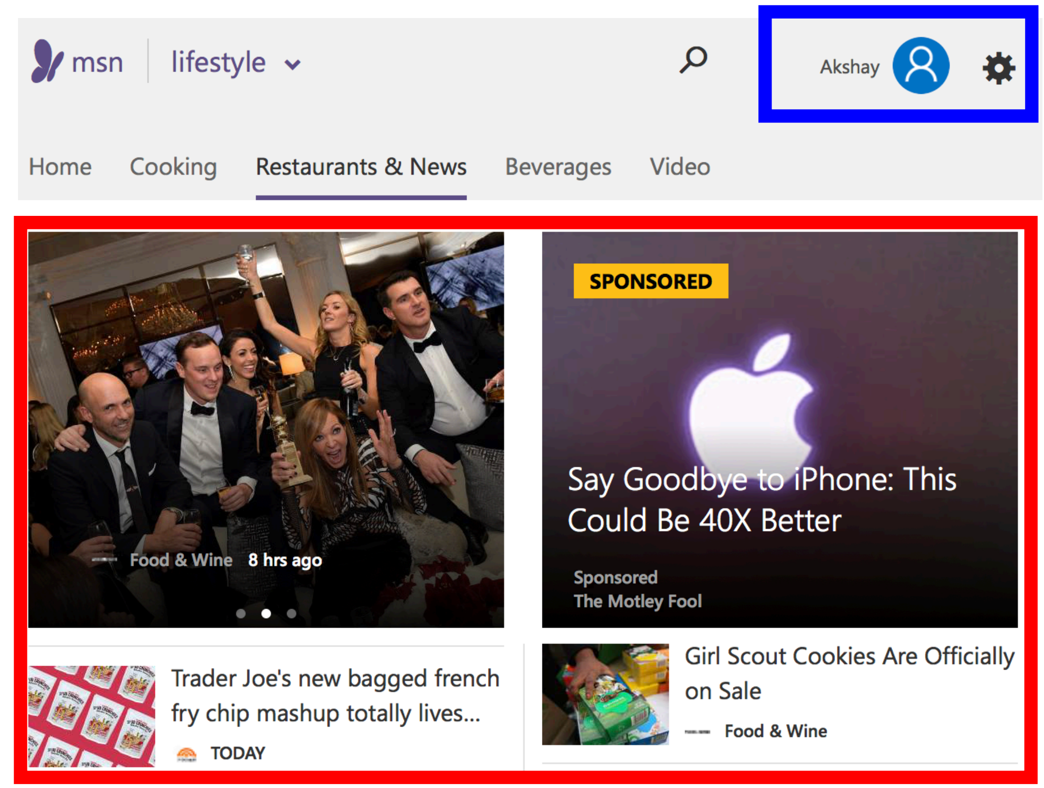
Fact: Can get $\sqrt{KT \log |\Pi|}$ regret.

Contextual Bandits

On each of T rounds:

1. Observe context x_t
2. Play action a_t
3. Observe reward $r_t(a_t, x_t)$

K = number of actions



$r_t = \#\ \text{clicks}$

$$\text{Regret}(T, \Pi) = \max_{\pi \in \Pi} \text{Reward}(T, \pi) - \text{LearnerReward}(T)$$

Fact: Can get $\sqrt{KT \log |\Pi|}$ regret.

A/B testing gets $(|\Pi|)^{1/3} T^{2/3}$

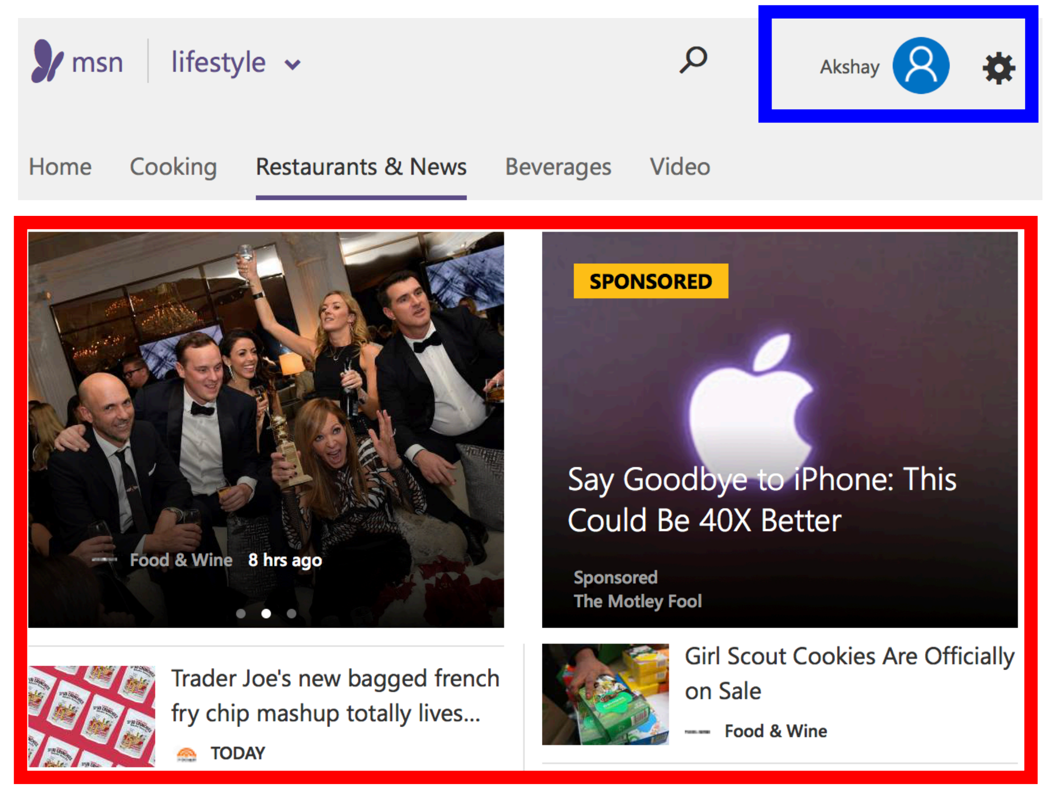
Offline Eval gets $(K \log |\Pi|)^{1/3} T^{2/3}$

Contextual Bandits

On each of T rounds:

1. Observe context x_t
2. Play action a_t
3. Observe reward $r_t(a_t, x_t)$

K = number of actions



$r_t = \#\text{ clicks}$

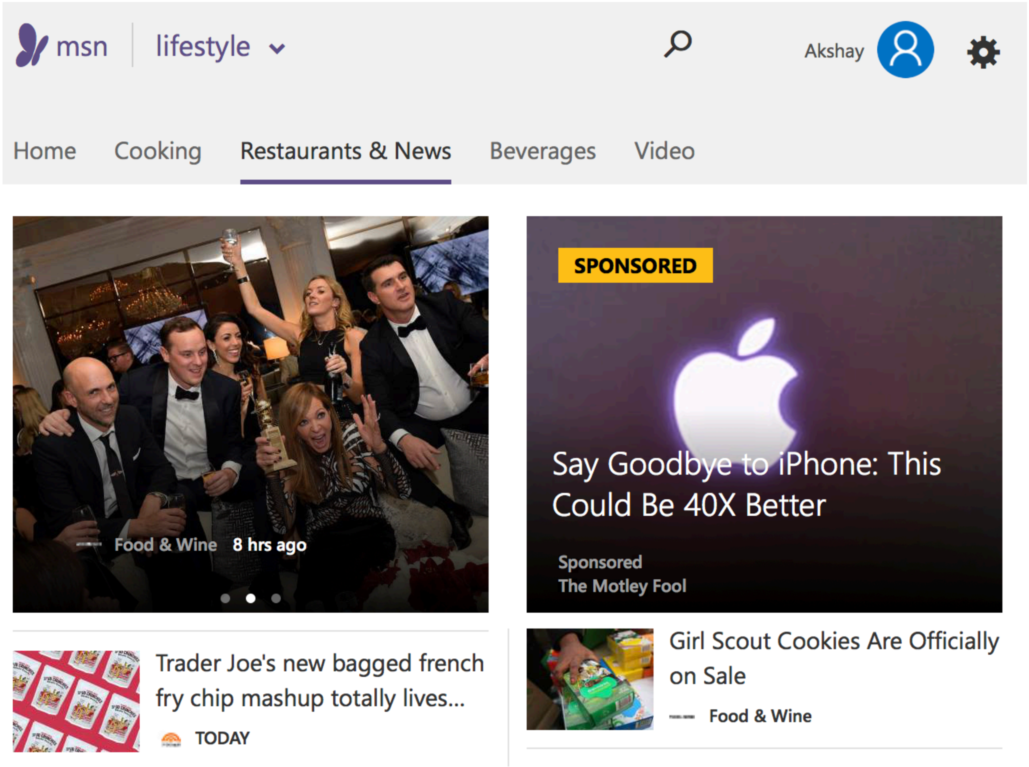
$$\text{Regret}(T, \Pi) = \max_{\pi \in \Pi} \text{Reward}(T, \pi) - \text{LearnerReward}(T)$$

A/B testing gets $(|\Pi|)^{1/3} T^{2/3}$

Fact: Can get $\sqrt{KT \log |\Pi|}$ regret. Offline Eval gets $(K \log |\Pi|)^{1/3} T^{2/3}$

Exponential with combinatorial action space!

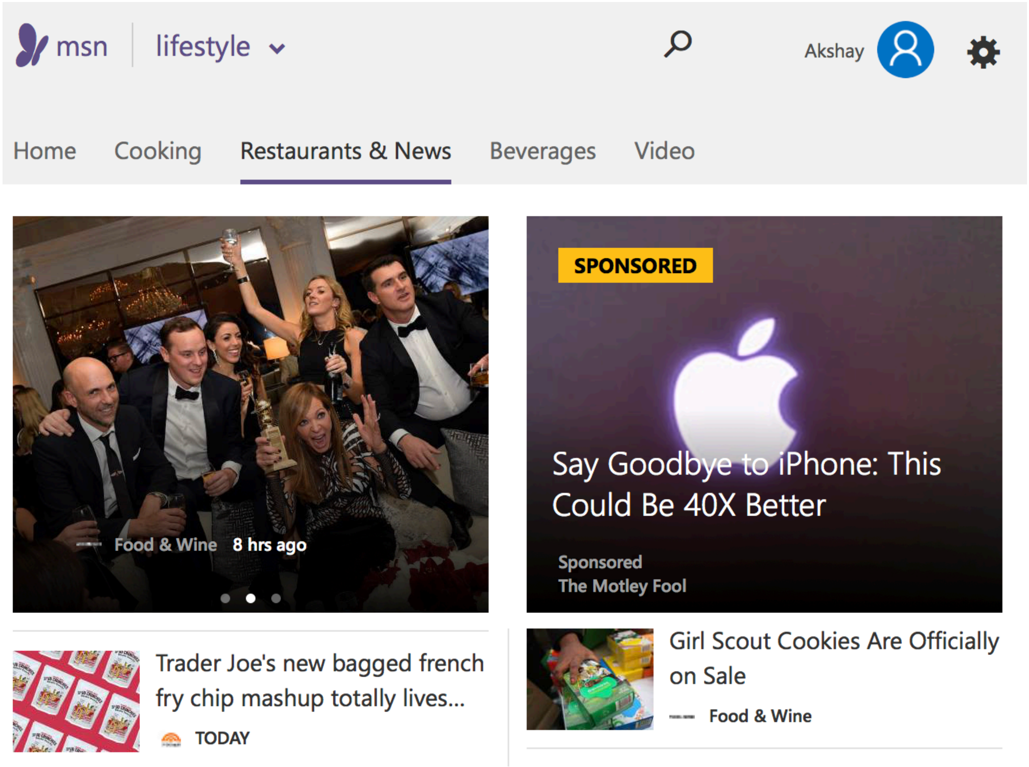
Contextual Semibandits



Contextual Semibandits

On each of T rounds:

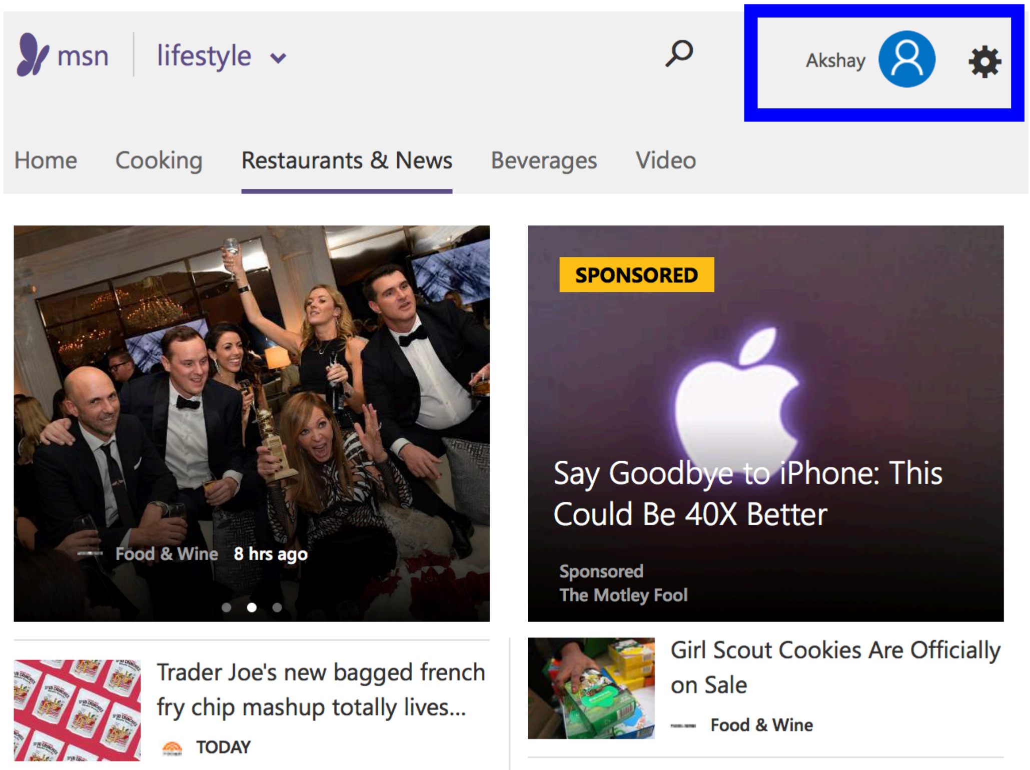
1. Observe context
2. Play action
3. Observe features
4. Observe reward



Contextual Semibandits

On each of T rounds:

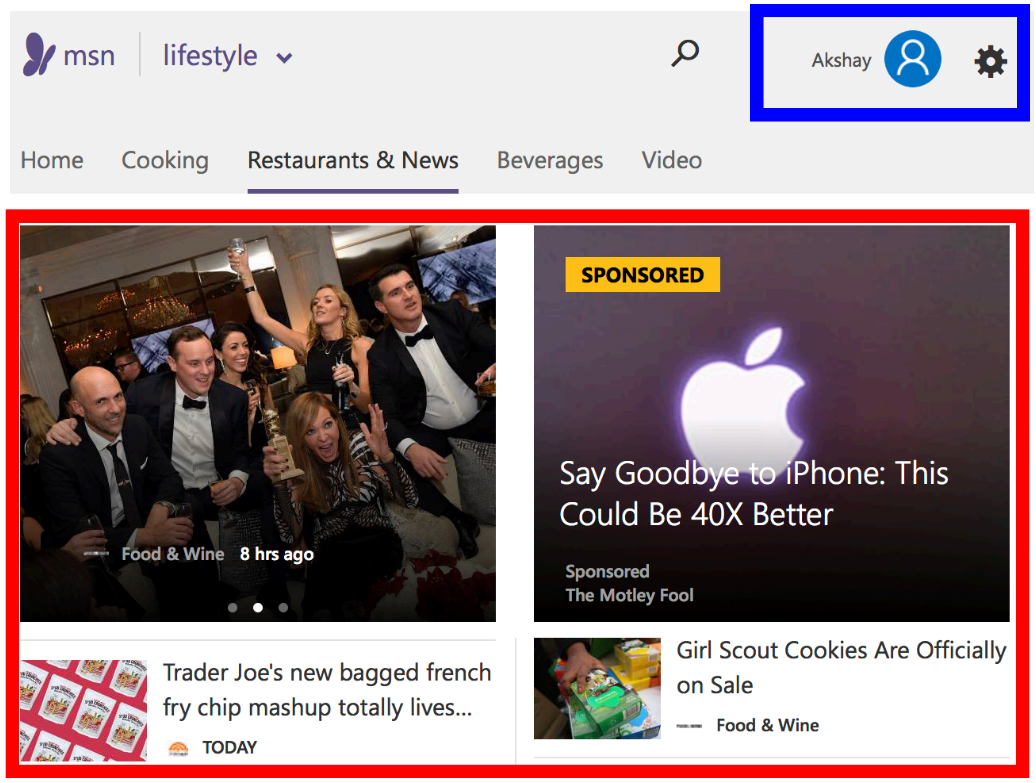
1. Observe context x_t
2. Play action
3. Observe features
4. Observe reward



Contextual Semibandits

On each of T rounds:

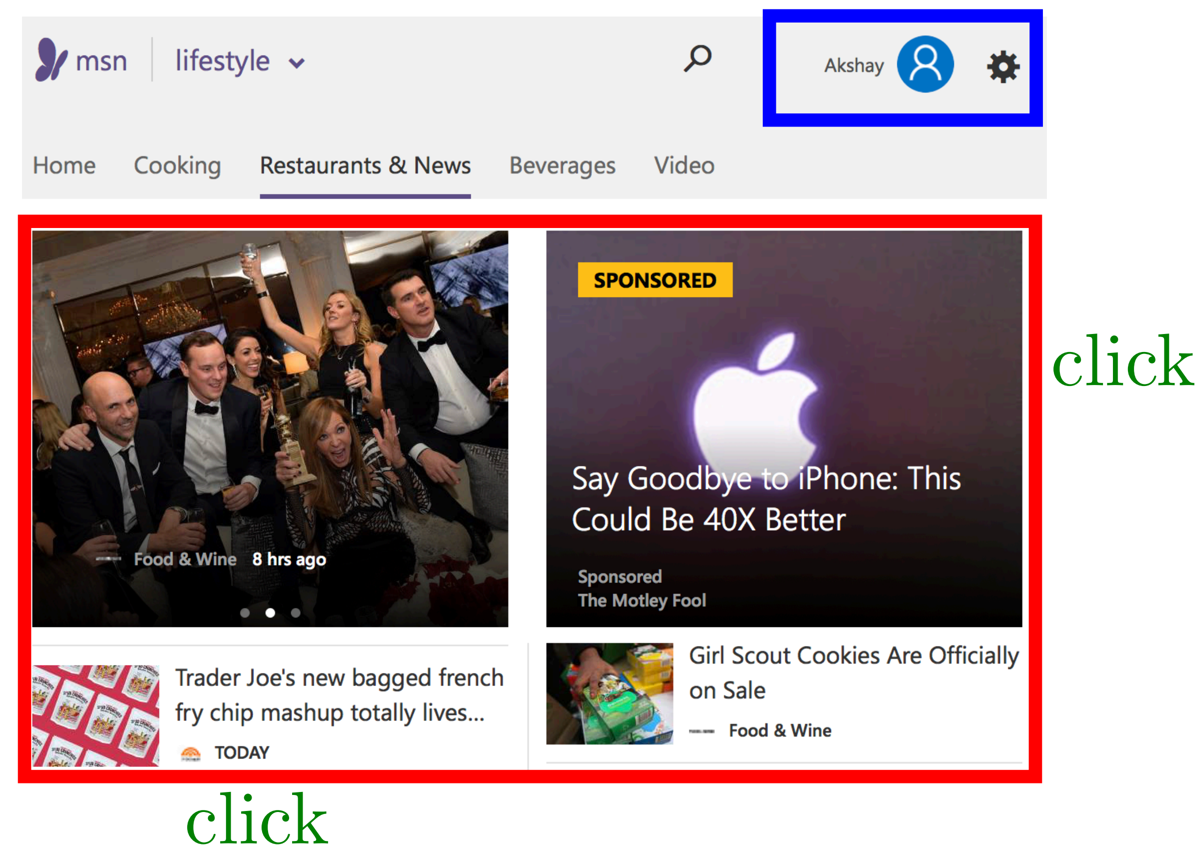
1. Observe context x_t
2. Play action $A_t = (a_1, \ldots, a_L)$
3. Observe features
4. Observe reward



Contextual Semibandits

On each of T rounds:

1. Observe context x_t
2. Play action $A_t = (a_1, \ldots, a_L)$
3. Observe features $\{y(a_\ell)\}_{\ell=1}^{L}$
4. Observe reward

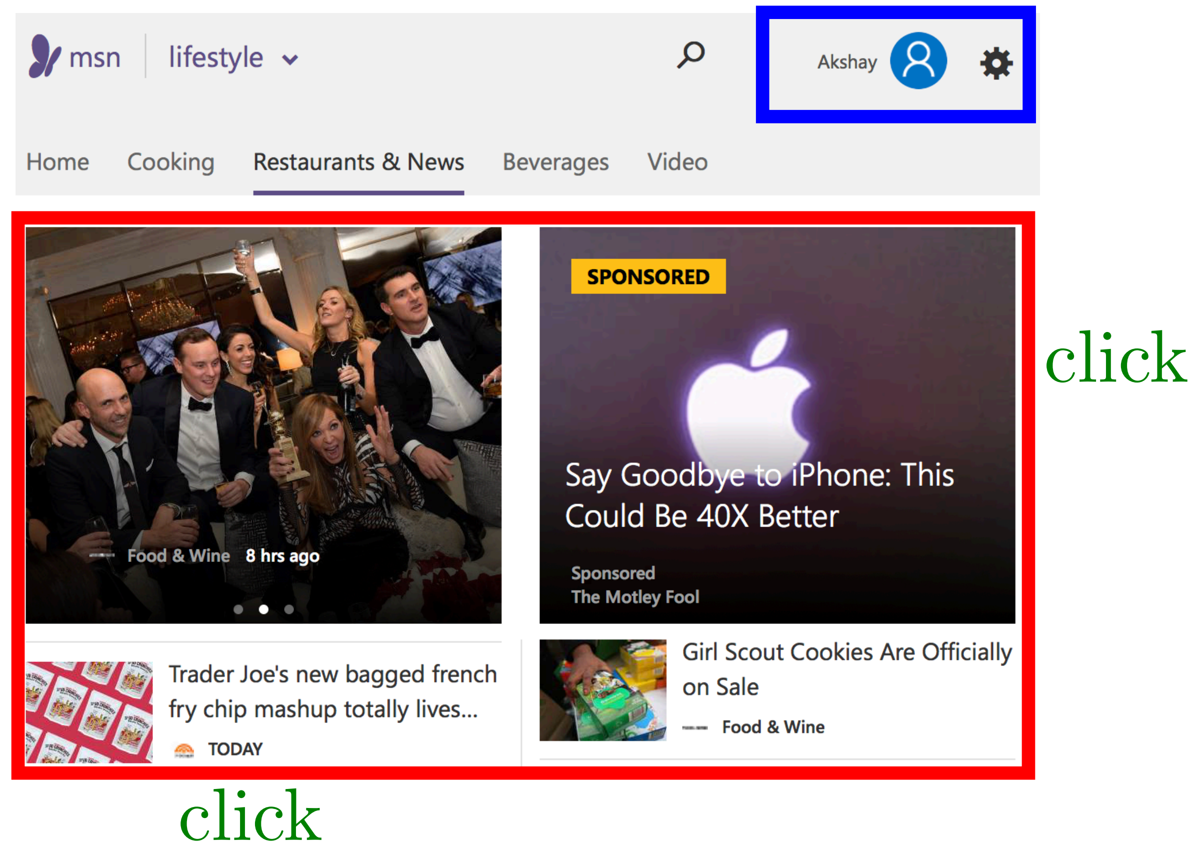


Contextual Semibandits

On each of T rounds:

1. Observe context x_t
2. Play action $A_t = (a_1, \ldots, a_L)$
3. Observe features $\{y(a_\ell)\}_{\ell=1}^{L}$
4. Observe reward

$$r_t(A_t, x_t) = \sum_{\ell} y(a_\ell) + \text{noise}$$



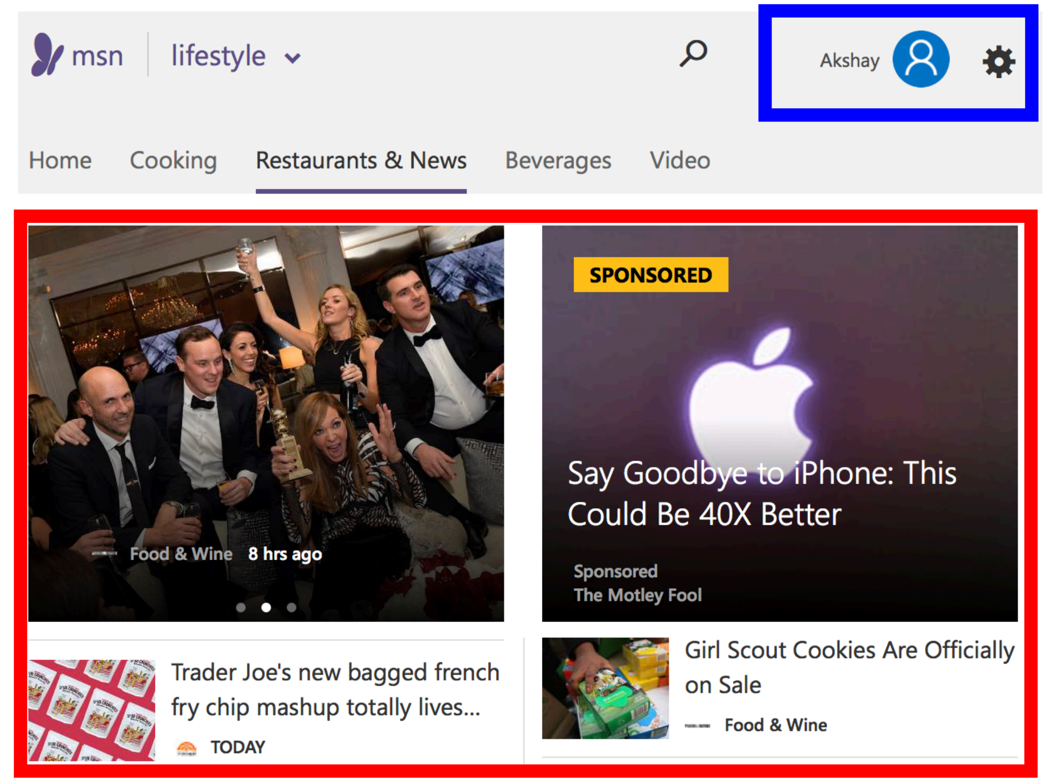
Contextual Semibandits

On each of T rounds:

1. Observe context x_t
2. Play action $A_t = (a_1, \ldots, a_L)$
3. Observe features $\{y(a_\ell)\}_{\ell=1}^{L}$
4. Observe reward

$$r_t(A_t, x_t) = \sum_{\ell} y(a_\ell) + \text{noise}$$

B = number of simple actions
L = composite action length



click

click

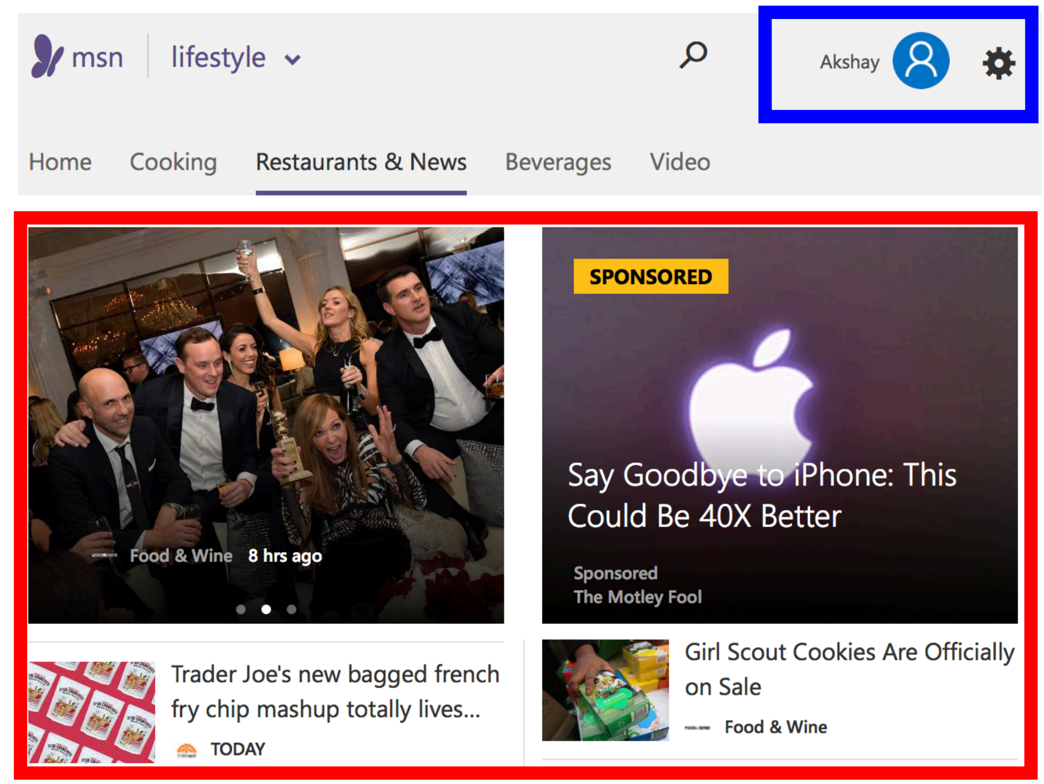
Contextual Semibandits

On each of T rounds:

1. Observe context x_t
2. Play action $A_t = (a_1, \ldots, a_L)$
3. Observe features $\{y(a_\ell)\}_{\ell=1}^{L}$
4. Observe reward

$$r_t(A_t, x_t) = \sum_\ell y(a_\ell) + \text{noise}$$

B = number of simple actions
L = composite action length



click

click

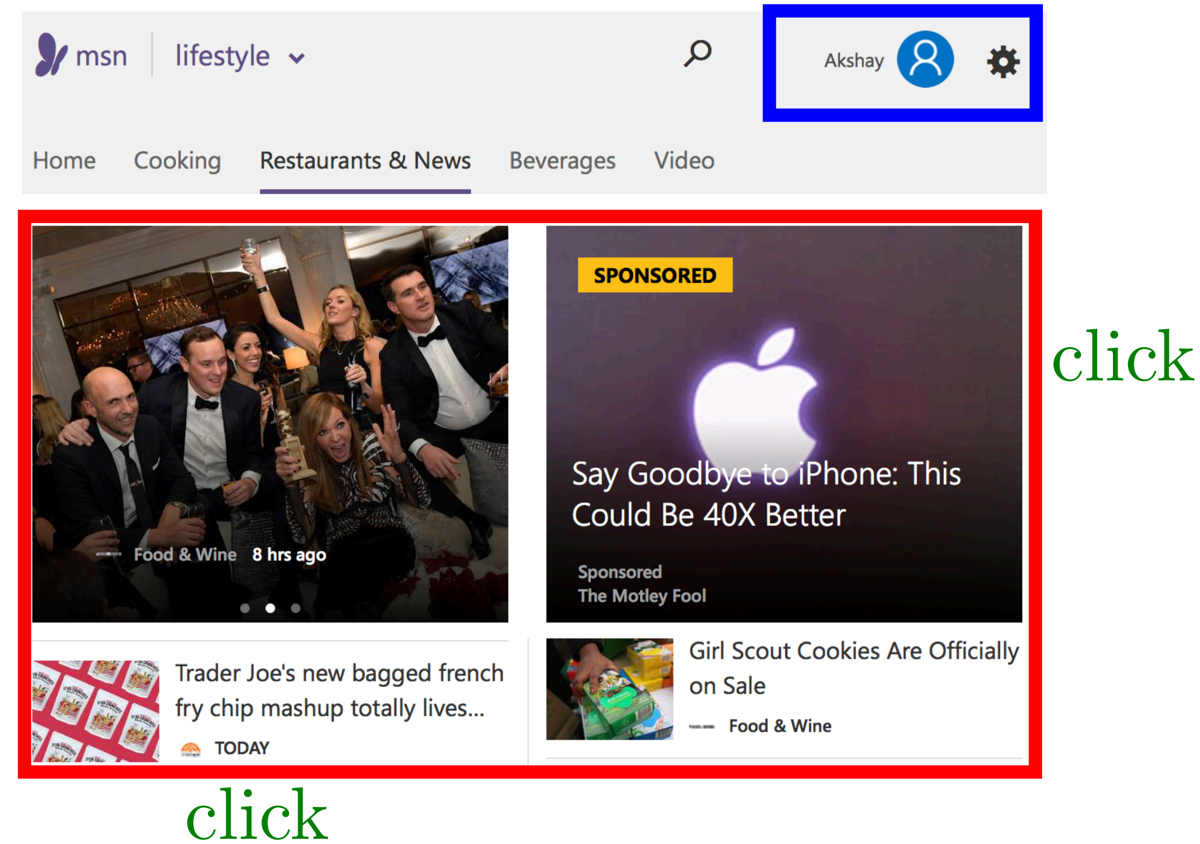
Question: Improve performance by leveraging reward structure + additional feedback?

Contextual Semibandits

On each of T rounds:

1. Observe context x_t
2. Play action $A_t = (a_1, \ldots, a_L)$
3. Observe features $\{y(a_\ell)\}_{\ell=1}^{L}$
4. Observe reward

$$r_t(A_t, x_t) = \sum_{\ell} y(a_\ell) + \text{noise}$$



B = number of simple actions
L = composite action length

Question: Improve performance by leveraging reward structure + additional feedback?

Challenges:

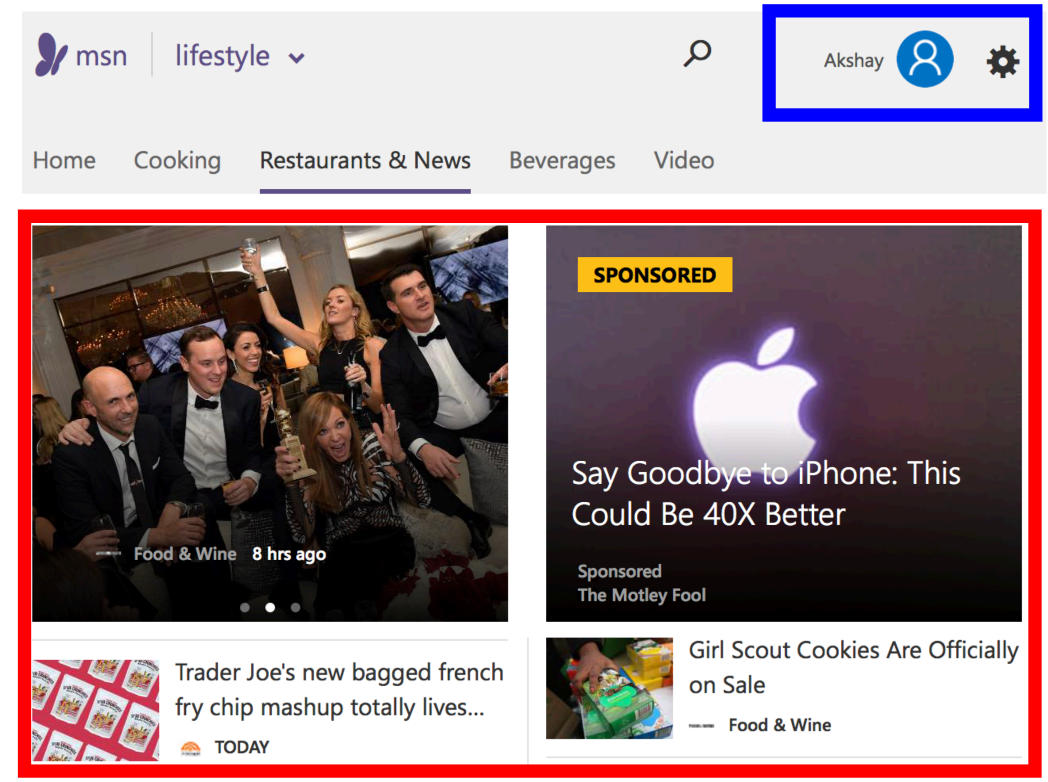
Contextual Semibandits

On each of T rounds:

1. Observe context x_t
2. Play action $A_t = (a_1, \ldots, a_L)$
3. Observe features $\{y(a_\ell)\}_{\ell=1}^{L}$
4. Observe reward

$$r_t(A_t, x_t) = \sum_{\ell} y(a_\ell) + \text{noise}$$

B = number of simple actions
L = composite action length



Question: Improve performance by leveraging reward structure + additional feedback?

Challenges:

- Off-policy evaluation?

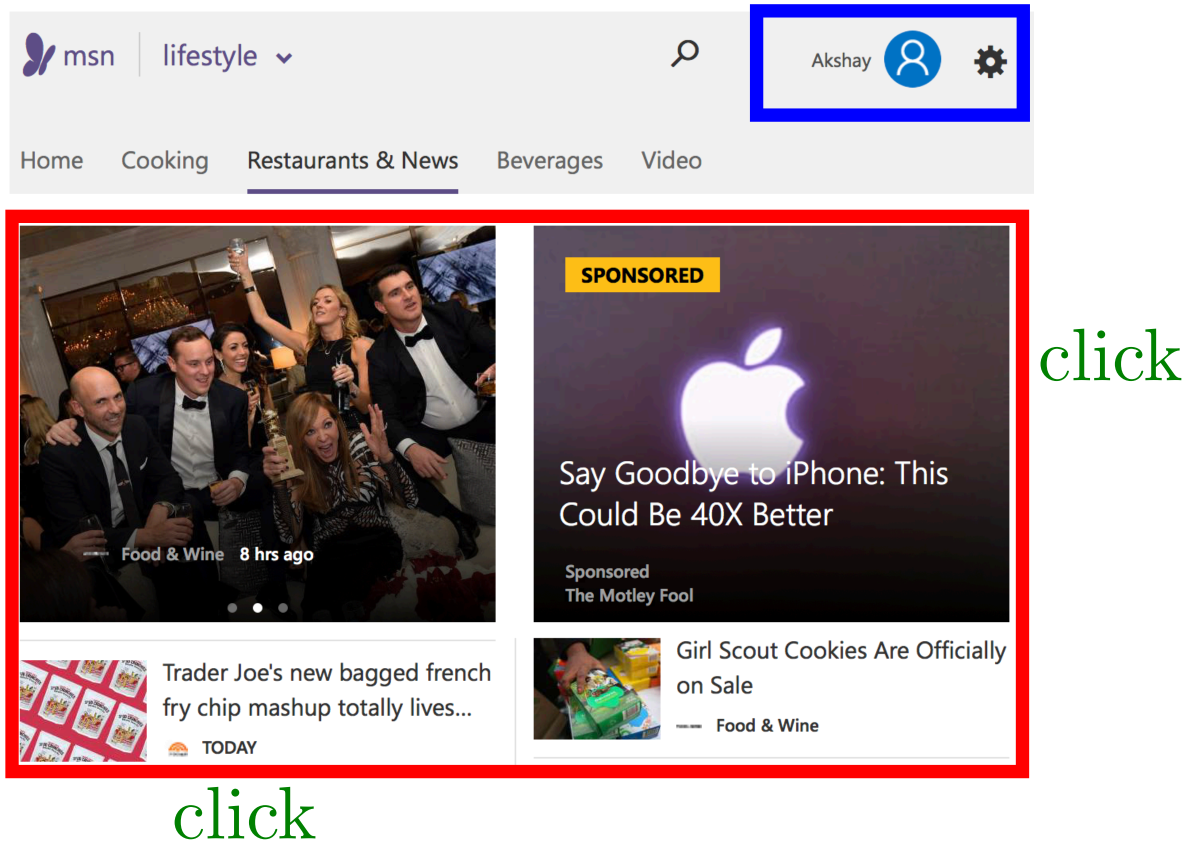
Contextual Semibandits

On each of T rounds:

1. Observe context x_t
2. Play action $A_t = (a_1, \ldots, a_L)$
3. Observe features $\{y(a_\ell)\}_{\ell=1}^{L}$
4. Observe reward

$$r_t(A_t, x_t) = \sum_{\ell} y(a_\ell) + \text{noise}$$

B = number of simple actions
L = composite action length



Question: Improve performance by leveraging reward structure + additional feedback?

Challenges:

- Off-policy evaluation?
- Explore vs Exploit?

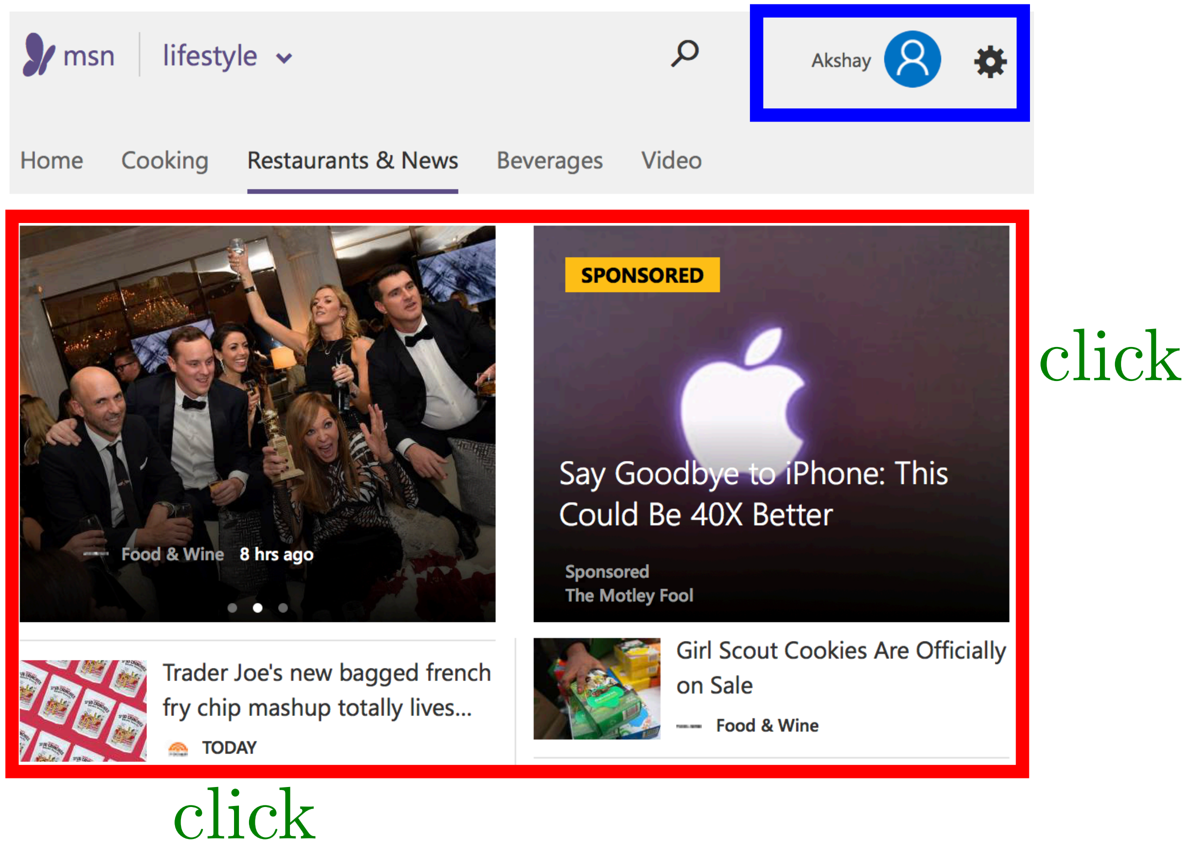
Contextual Semibandits

On each of T rounds:

1. Observe context x_t
2. Play action $A_t = (a_1, \ldots, a_L)$
3. Observe features $\{y(a_\ell)\}_{\ell=1}^{L}$
4. Observe reward

$$r_t(A_t, x_t) = \sum_\ell y(a_\ell) + \text{noise}$$

B = number of simple actions
L = composite action length



Question: Improve performance by leveraging reward structure + additional feedback?

Challenges:

- Off-policy evaluation?
- Explore vs Exploit?
- Computational Efficiency?

Results

[Krishnamurthy, Agarwal, Dudik. NeurIPS 2016]

Results

Theorem: Efficient algorithm with $\sqrt{BT\log(|\Pi|)}$ regret

Parameters: T rounds, B simple actions, composite action length L

[Krishnamurthy, Agarwal, Dudik. NeurIPS 2016]

Results

Theorem: Efficient algorithm with $\sqrt{BT\log(|\Pi|)}$ regret

Parameters: T rounds, B simple actions, composite action length L

- Exponentially better than $\sqrt{B^L T\log(|\Pi|)}$ for naive contextual bandits
- Computationally efficient with rich policy classes

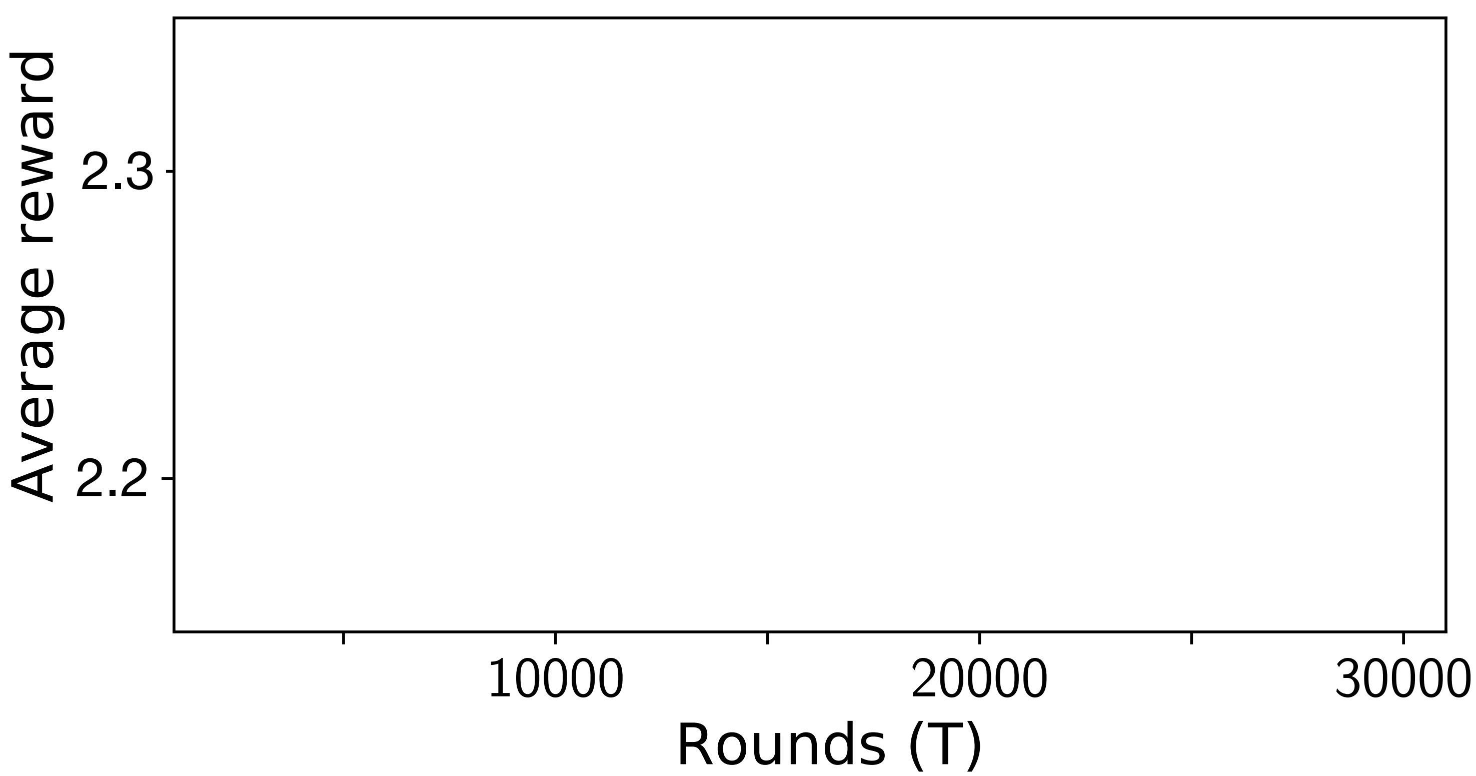
[Krishnamurthy, Agarwal, Dudik. NeurIPS 2016]

Results

Theorem: Efficient algorithm with $\sqrt{BT\log(|\Pi|)}$ regret

Parameters: T rounds, B simple actions, composite action length L

- Exponentially better than $\sqrt{B^L T\log(|\Pi|)}$ for naive contextual bandits
- Computationally efficient with rich policy classes



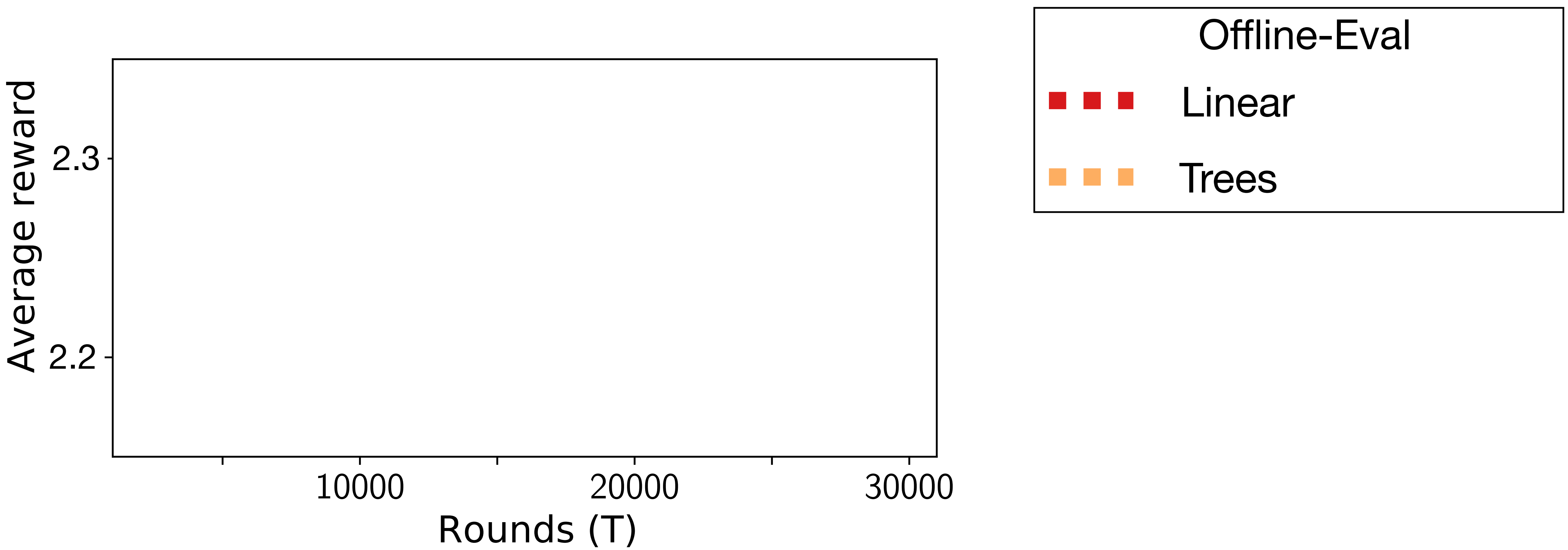
[Krishnamurthy, Agarwal, Dudik. NeurIPS 2016]

Results

Theorem: Efficient algorithm with $\sqrt{BT\log(|\Pi|)}$ regret

Parameters: T rounds, B simple actions, composite action length L

- Exponentially better than $\sqrt{B^L T\log(|\Pi|)}$ for naive contextual bandits
- Computationally efficient with rich policy classes



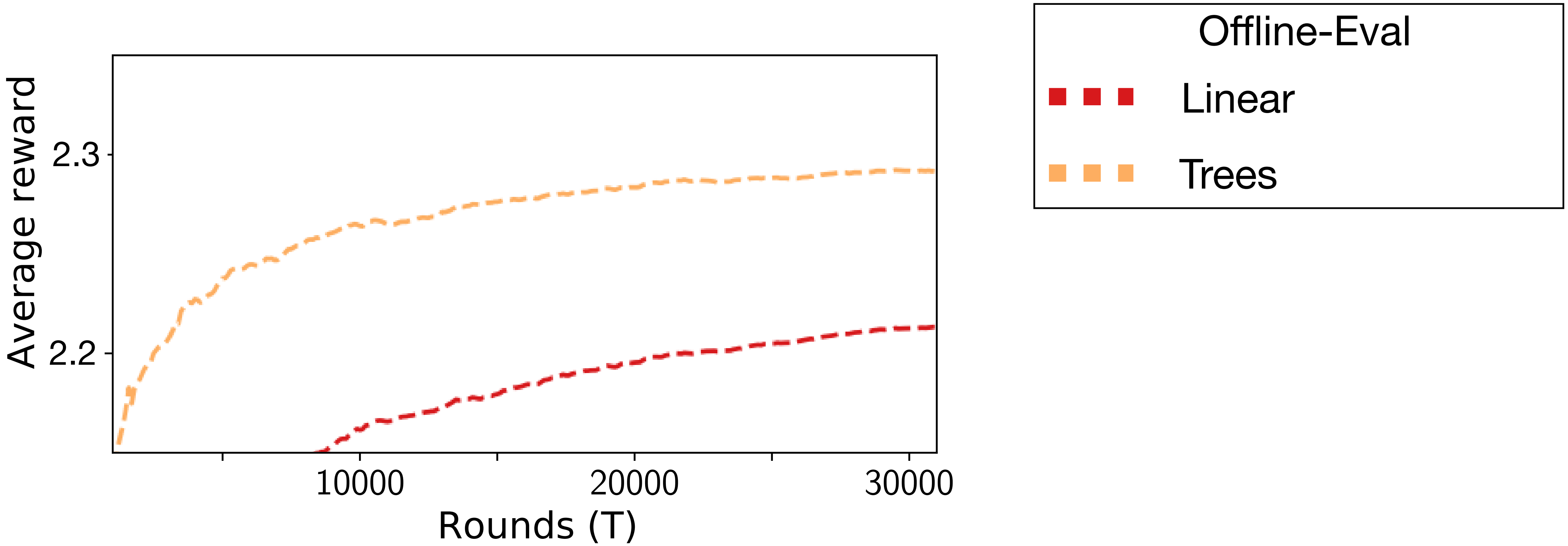
[Krishnamurthy, Agarwal, Dudik. NeurIPS 2016]

Results

Theorem: Efficient algorithm with $\sqrt{BT\log(|\Pi|)}$ regret

Parameters: T rounds, B simple actions, composite action length L

- Exponentially better than $\sqrt{B^L T\log(|\Pi|)}$ for naive contextual bandits
- Computationally efficient with rich policy classes



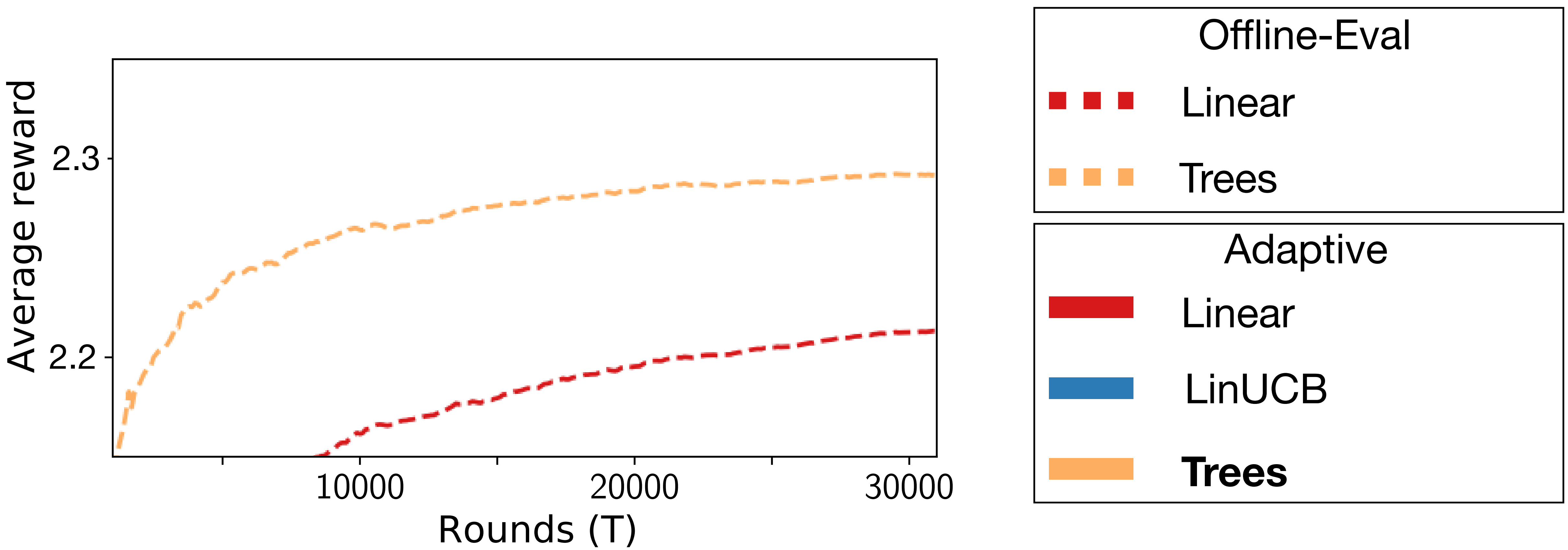
[Krishnamurthy, Agarwal, Dudik. NeurIPS 2016]

Results

Theorem: Efficient algorithm with $\sqrt{BT\log(|\Pi|)}$ regret

Parameters: T rounds, B simple actions, composite action length L

- Exponentially better than $\sqrt{B^L T\log(|\Pi|)}$ for naive contextual bandits
- Computationally efficient with rich policy classes



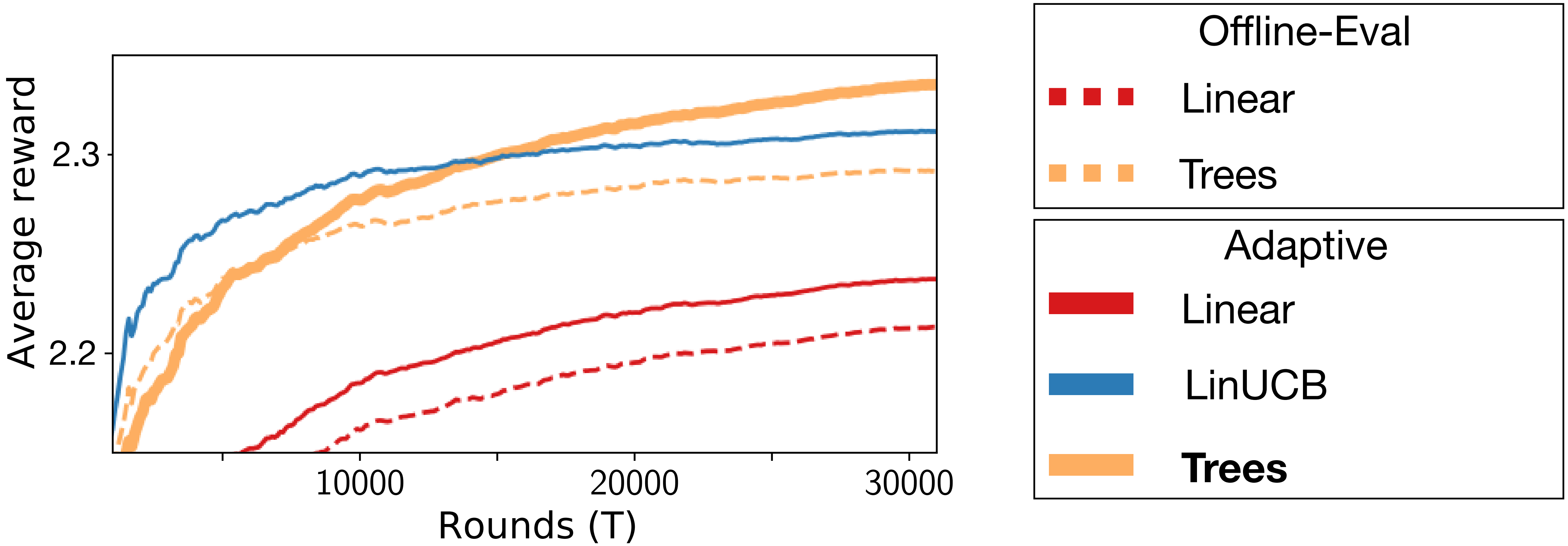
[Krishnamurthy, Agarwal, Dudik. NeurIPS 2016]

Results

Theorem: Efficient algorithm with $\sqrt{BT\log(|\Pi|)}$ regret

Parameters: T rounds, B simple actions, composite action length L

- Exponentially better than $\sqrt{B^L T\log(|\Pi|)}$ for naive contextual bandits
- Computationally efficient with rich policy classes



[Krishnamurthy, Agarwal, Dudik. NeurIPS 2016]

Techniques — Off-policy evaluation

Subproblem: Given data collected by a logging policy, estimate reward of a target policy

Techniques — Off-policy evaluation

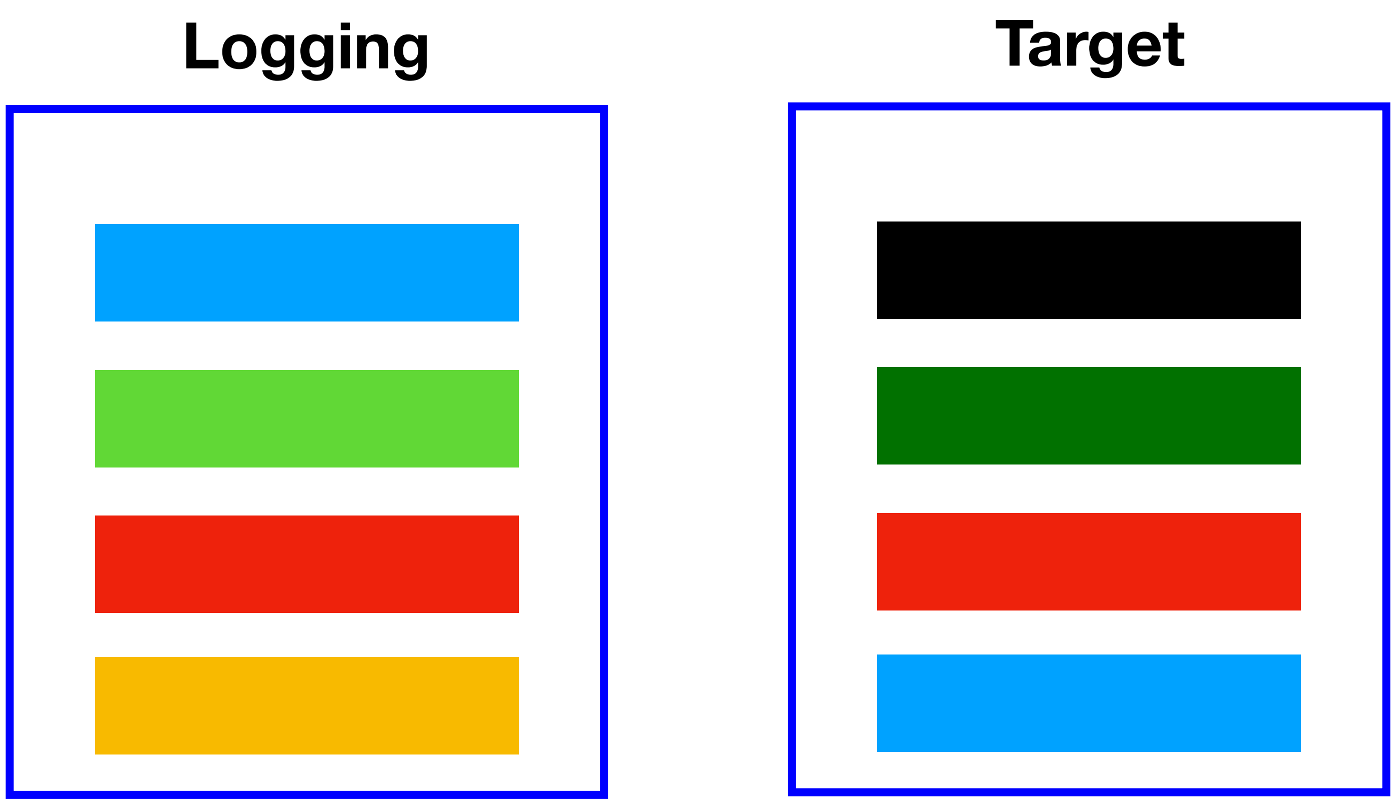
Subproblem: Given data collected by a logging policy, estimate reward of a target policy

Logging

Target

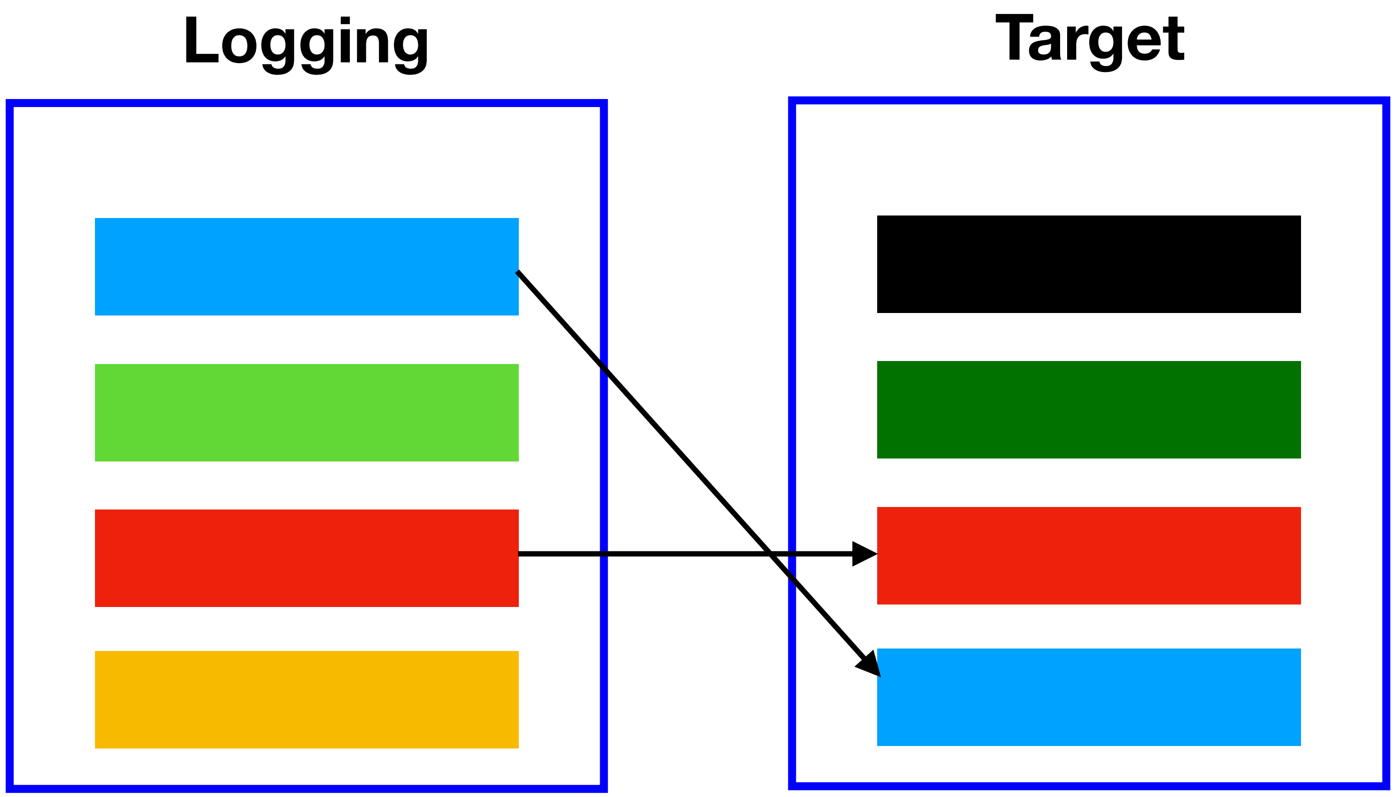
Techniques — Off-policy evaluation

Subproblem: Given data collected by a logging policy, estimate reward of a target policy



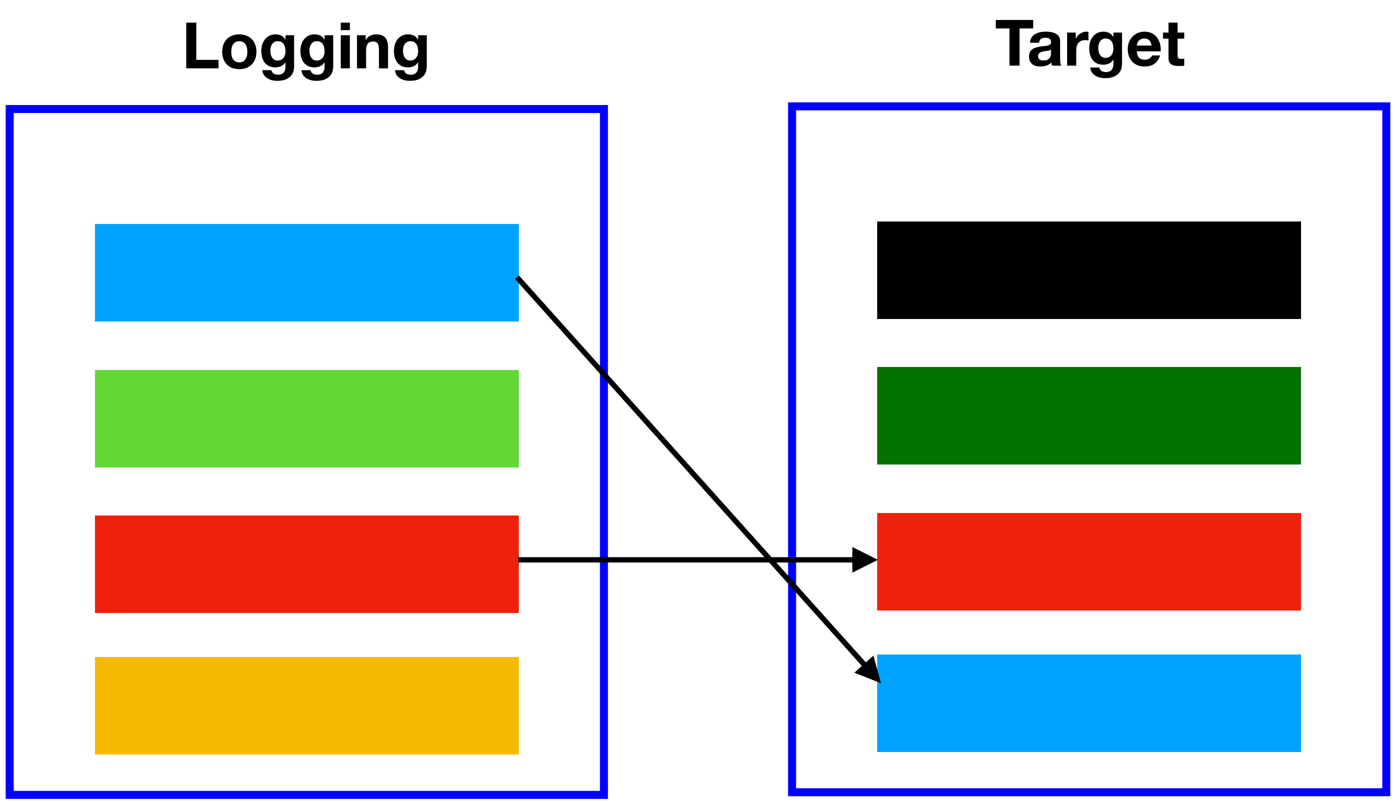
Techniques — Off-policy evaluation

Subproblem: Given data collected by a logging policy, estimate reward of a target policy



Techniques — Off-policy evaluation

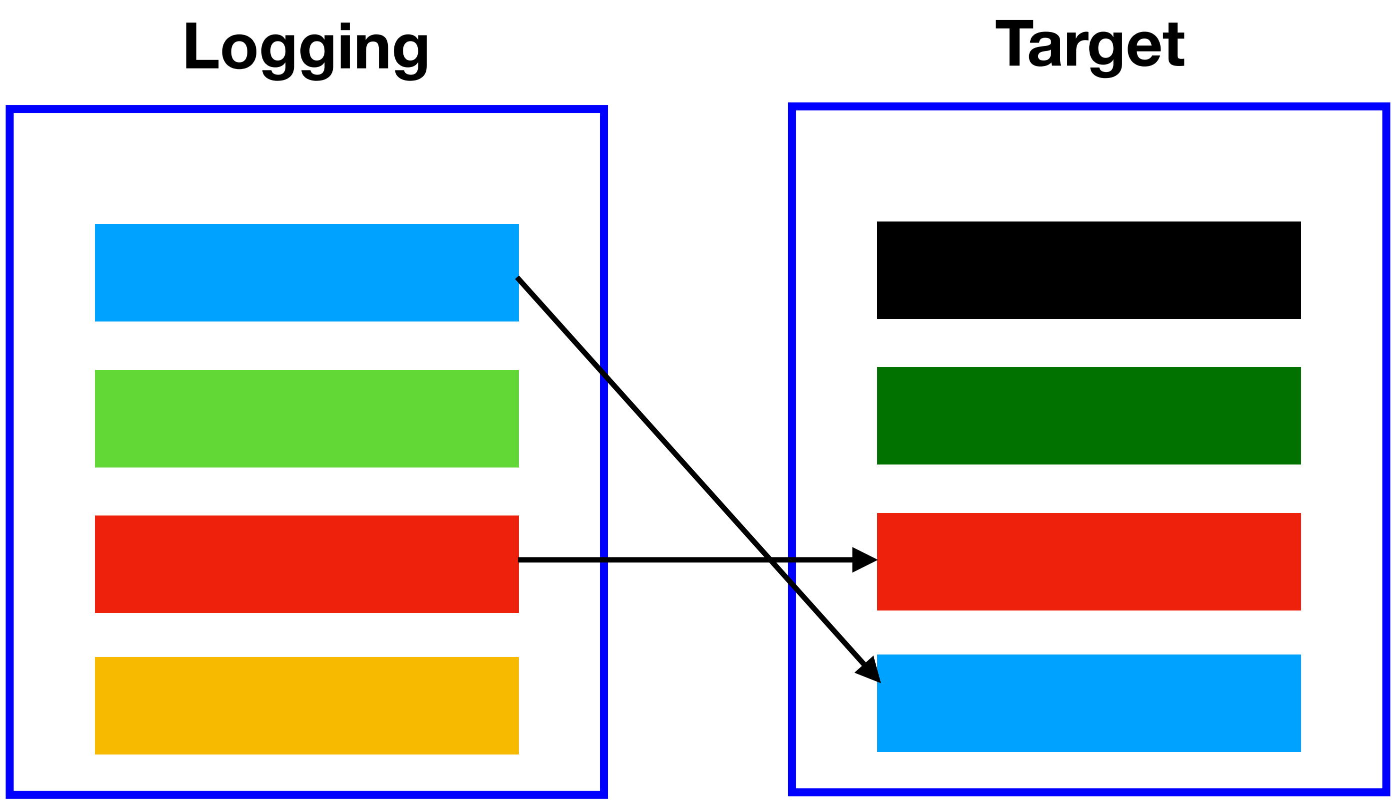
Subproblem: Given data collected by a logging policy, estimate reward of a target policy



Idea: Use partial matches!

Techniques — Off-policy evaluation

Subproblem: Given data collected by a logging policy, estimate reward of a target policy



Idea: Use partial matches!

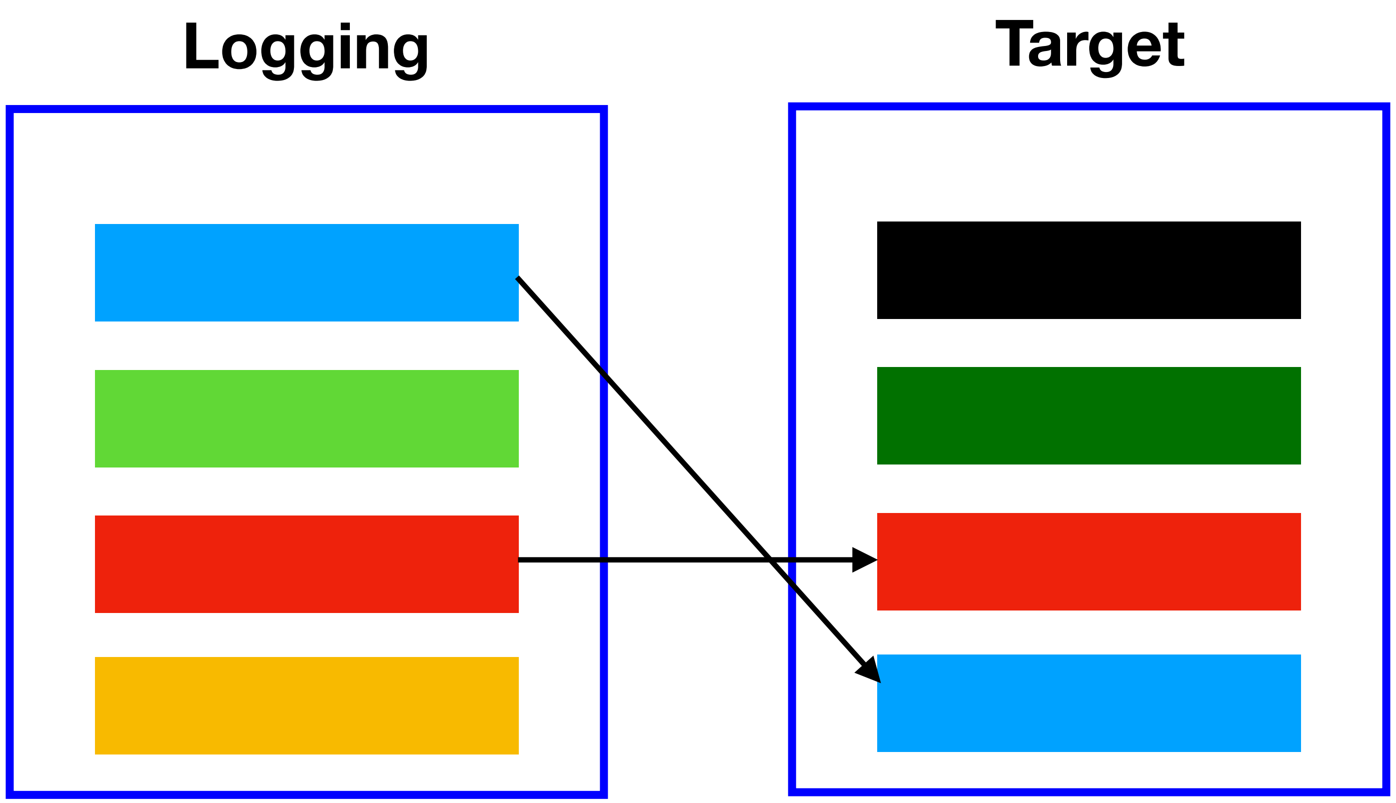
$$\text{If } A \sim Q(\cdot|x)$$

$$\hat{y}(a) = \frac{y(a)\mathbf{1}(a \in A)}{Q(a \in A|x)}$$

$$\hat{r}(\pi, x) = \sum_{a \in \pi(x)} \hat{y}(a)$$

Techniques — Off-policy evaluation

Subproblem: Given data collected by a logging policy, estimate reward of a target policy



Idea: Use partial matches!

$$\text{If } A \sim Q(\cdot|x)$$

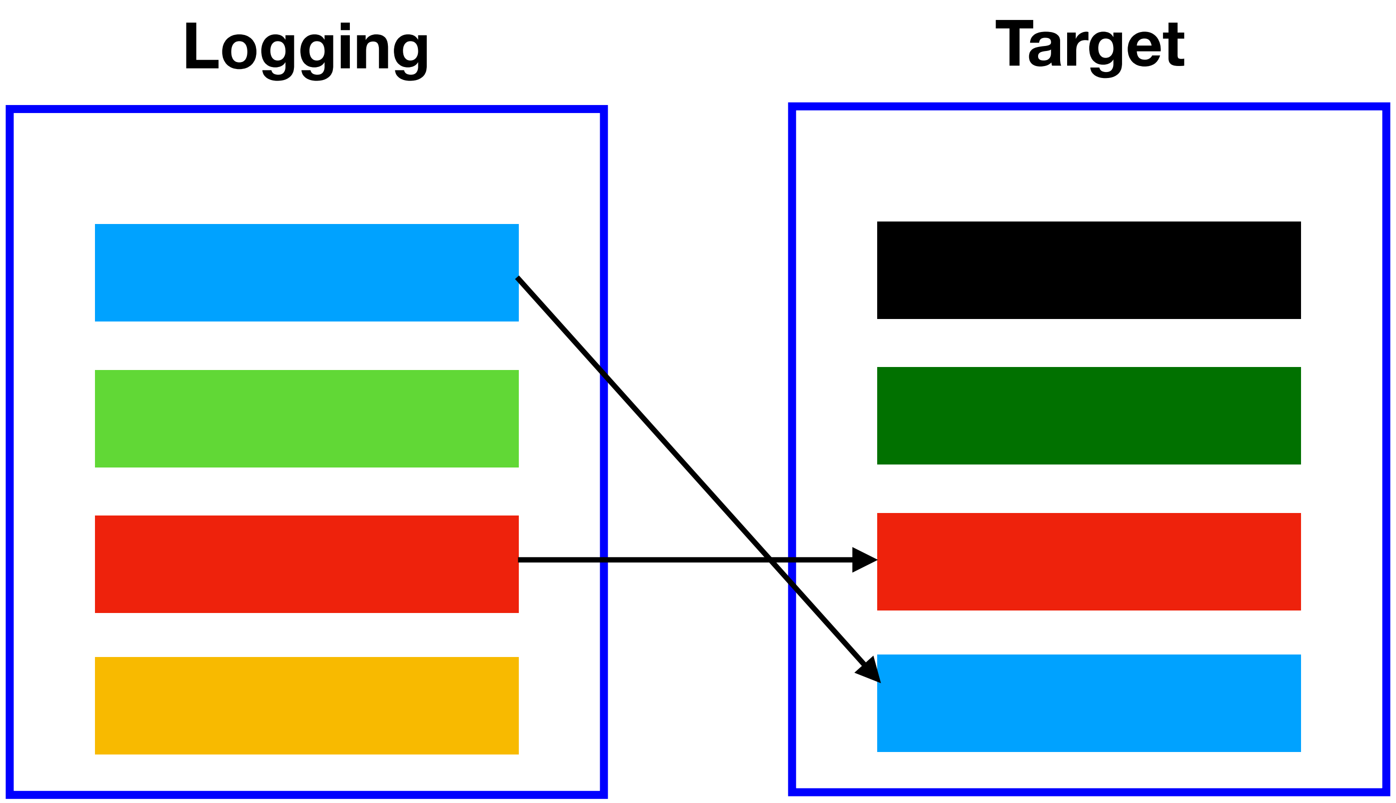
$$\hat{y}(a) = \frac{y(a)\mathbf{1}(a \in A)}{Q(a \in A|x)}$$

$$\hat{r}(\pi, x) = \sum_{a \in \pi(x)} \hat{y}(a)$$

- Uniform Q gives O(B) variance

Techniques — Off-policy evaluation

Subproblem: Given data collected by a logging policy, estimate reward of a target policy



Idea: Use partial matches!

$$\text{If } A \sim Q(\cdot|x)$$

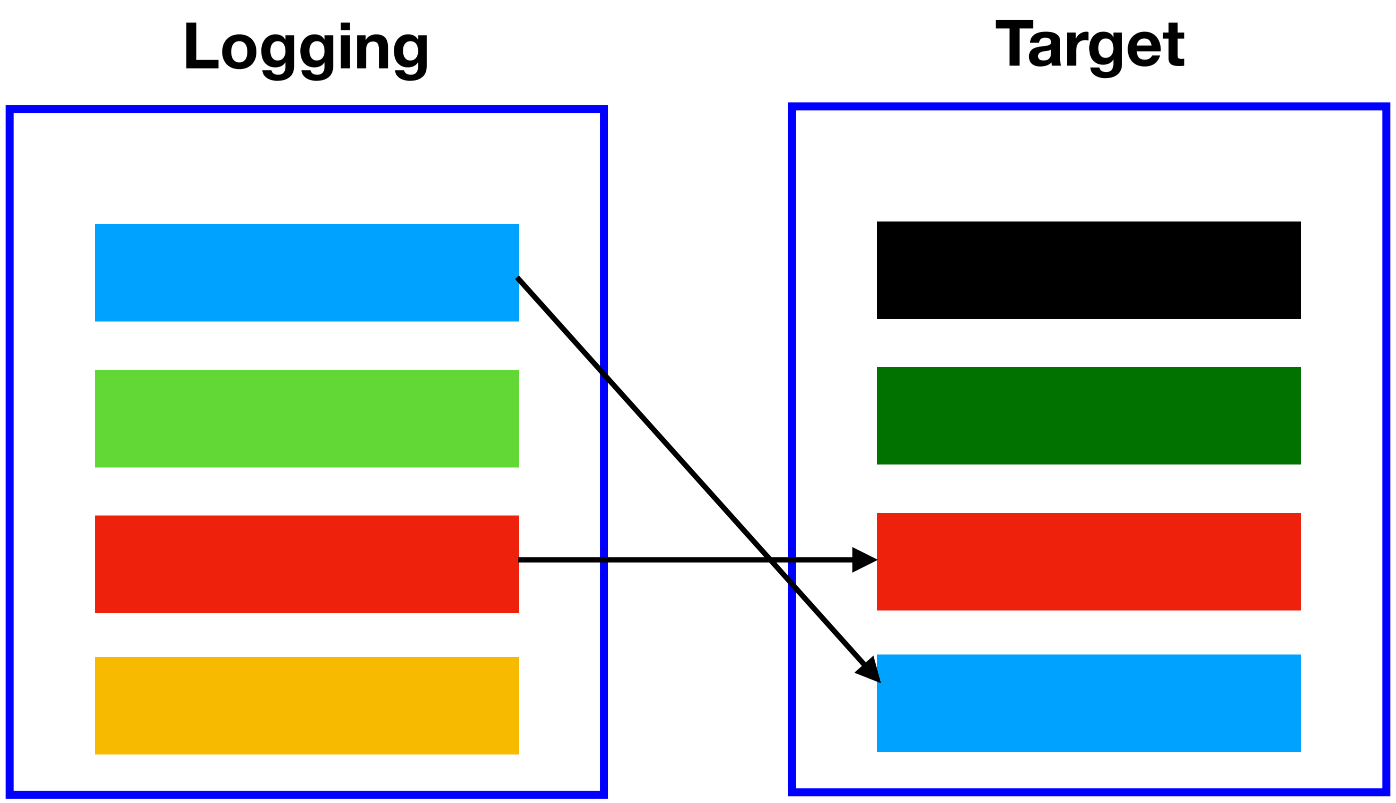
$$\hat{y}(a) = \frac{y(a)\mathbf{1}(a \in A)}{Q(a \in A|x)}$$

$$\hat{r}(\pi, x) = \sum_{a \in \pi(x)} \hat{y}(a)$$

- Uniform Q gives O(B) variance
- Immediately gives decent algorithm (eps-greedy)

Techniques — Off-policy evaluation

Subproblem: Given data collected by a logging policy, estimate reward of a target policy



Idea: Use partial matches!

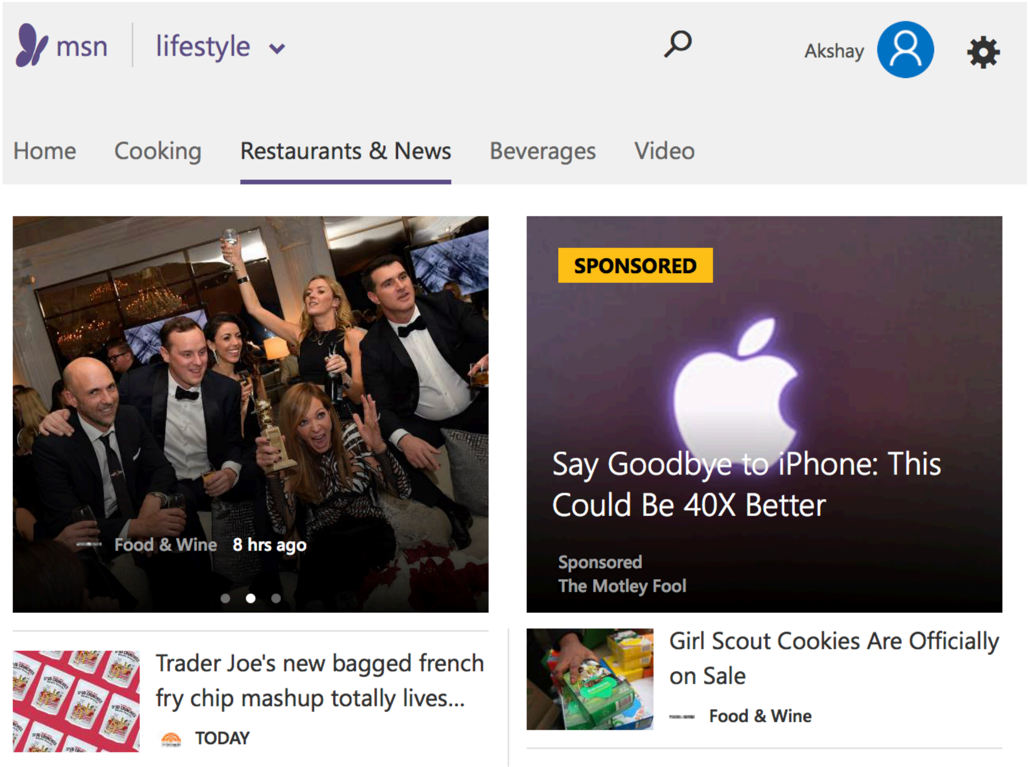
If $A \sim Q(\cdot|x)$

$$\hat{y}(a) = \frac{y(a)\mathbf{1}(a \in A)}{Q(a \in A|x)}$$

$$\hat{r}(\pi, x) = \sum_{a \in \pi(x)} \hat{y}(a)$$

- Uniform Q gives O(B) variance
- Immediately gives decent algorithm (eps-greedy)
- We need more refined approach

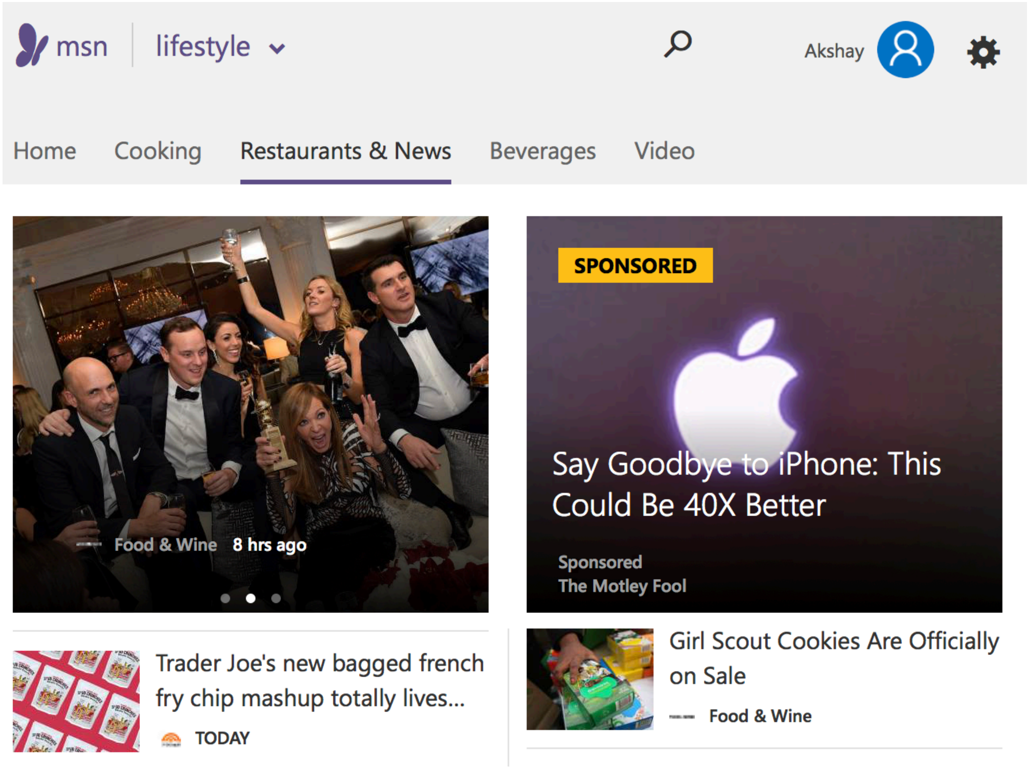
Combinatorial Contextual Bandits



Combinatorial Contextual Bandits

On each of T rounds:

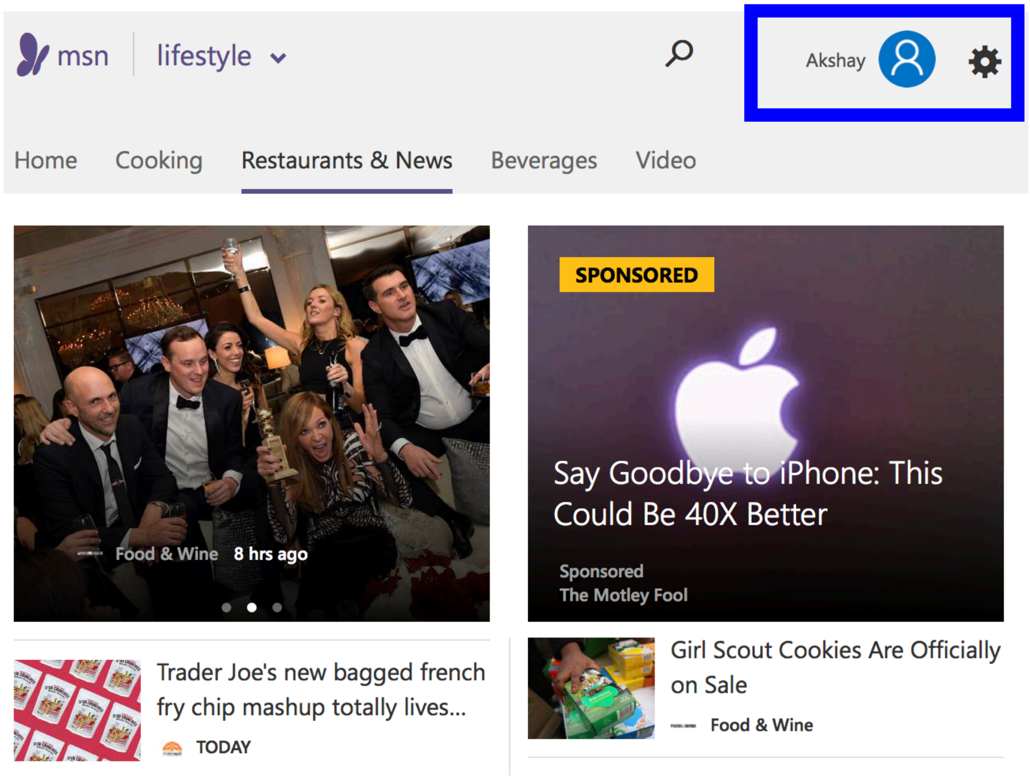
1. Observe context
2. Play action
3. *Unobserved* features
4. Observe reward



Combinatorial Contextual Bandits

On each of T rounds:

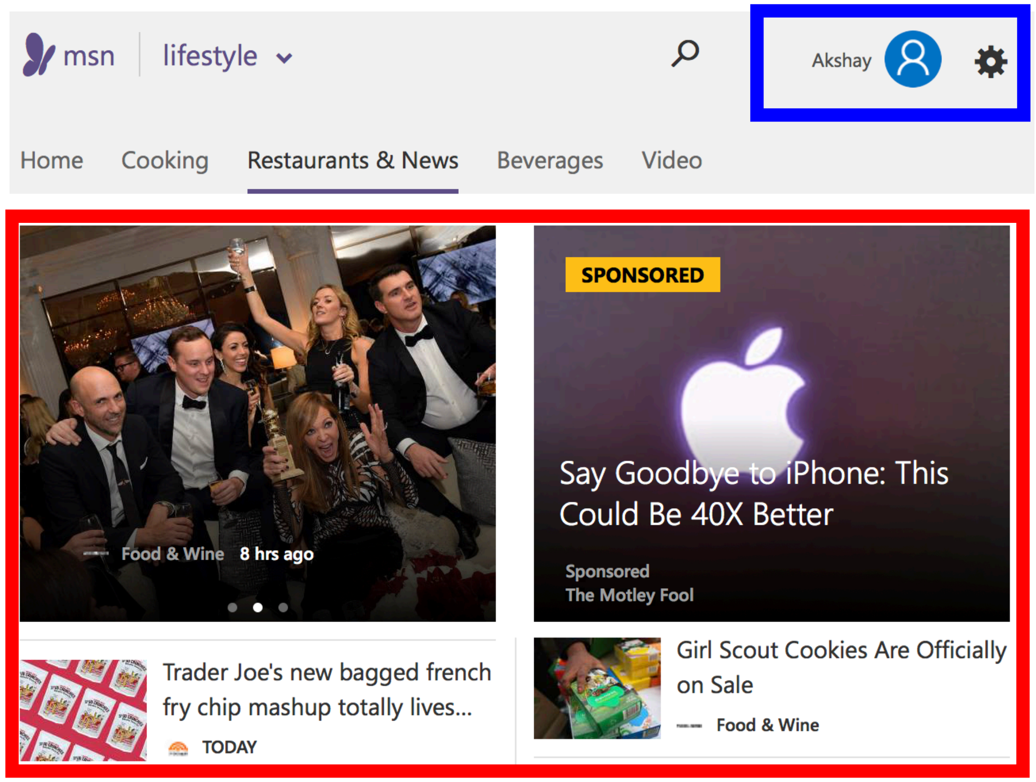
1. Observe context x_t
2. Play action
3. *Unobserved* features
4. Observe reward



Combinatorial Contextual Bandits

On each of T rounds:

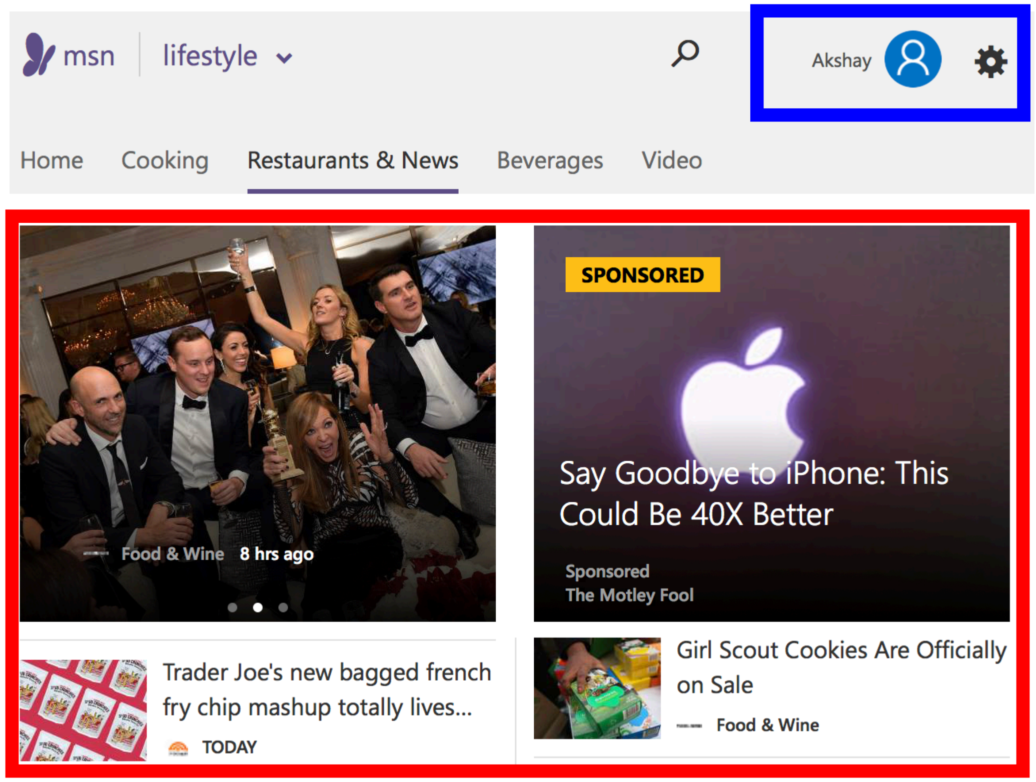
1. Observe context x_t
2. Play action $A_t = (a_1, \ldots, a_L)$
3. *Unobserved* features
4. Observe reward



Combinatorial Contextual Bandits

On each of T rounds:

1. Observe context x_t
2. Play action $A_t = (a_1, \ldots, a_L)$
3. *Unobserved* features $\{y(\ell, a_\ell)\}_{\ell=1}^{L}$
4. Observe reward

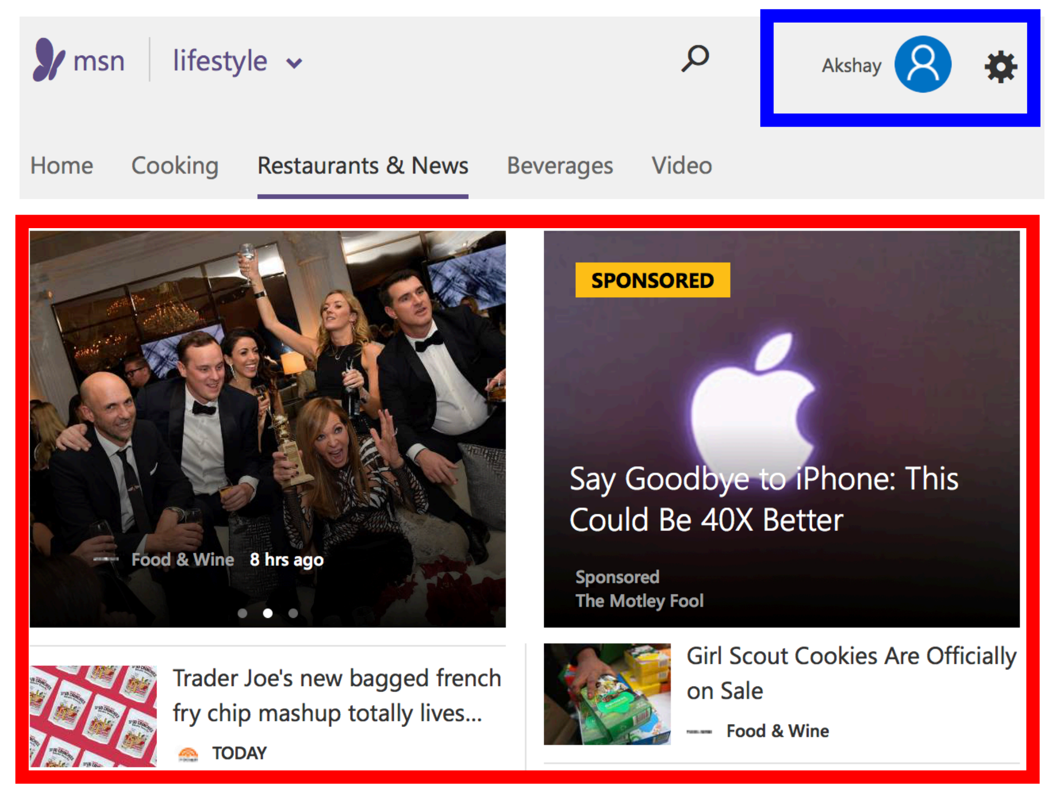


Combinatorial Contextual Bandits

On each of T rounds:

1. Observe context x_t
2. Play action $A_t = (a_1, \ldots, a_L)$
3. *Unobserved* features $\{y(\ell, a_\ell)\}_{\ell=1}^{L}$
4. Observe reward

$$r_t(A_t, x_t) = \sum_{\ell} y(\ell, a_\ell) + \text{noise}$$



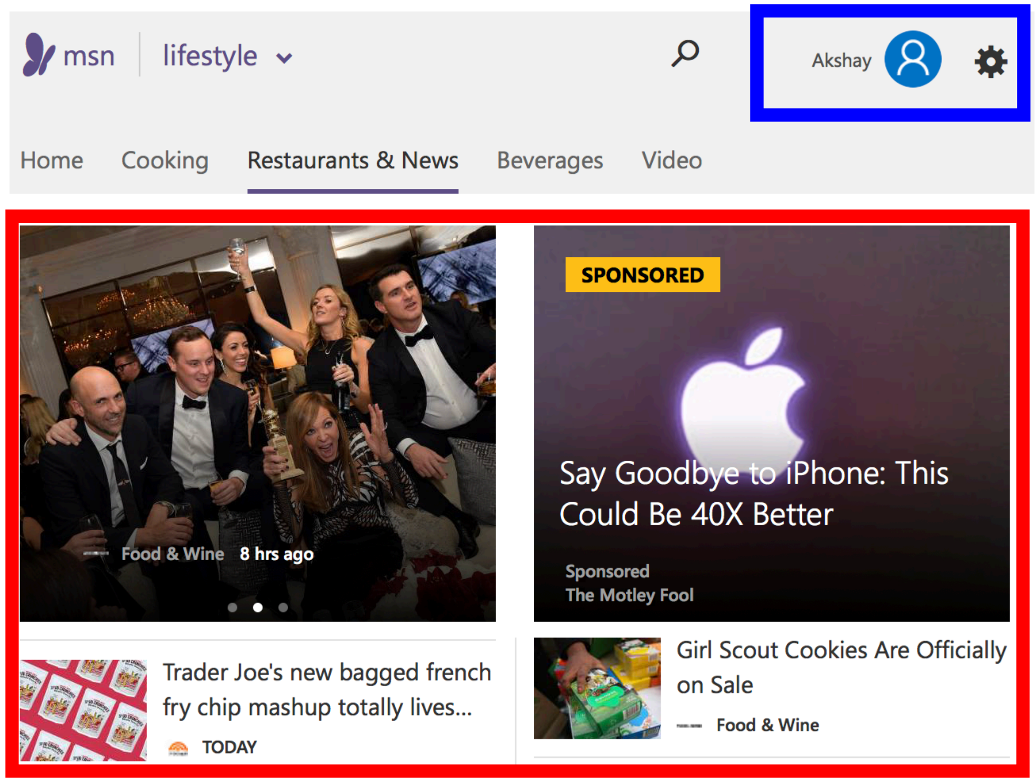
Combinatorial Contextual Bandits

On each of T rounds:

1. Observe context x_t
2. Play action $A_t = (a_1, \ldots, a_L)$
3. *Unobserved* features $\{y(\ell, a_\ell)\}_{\ell=1}^{L}$
4. Observe reward

$$r_t(A_t, x_t) = \sum_\ell y(\ell, a_\ell) + \text{noise}$$

B = number of simple actions
L = composite action length

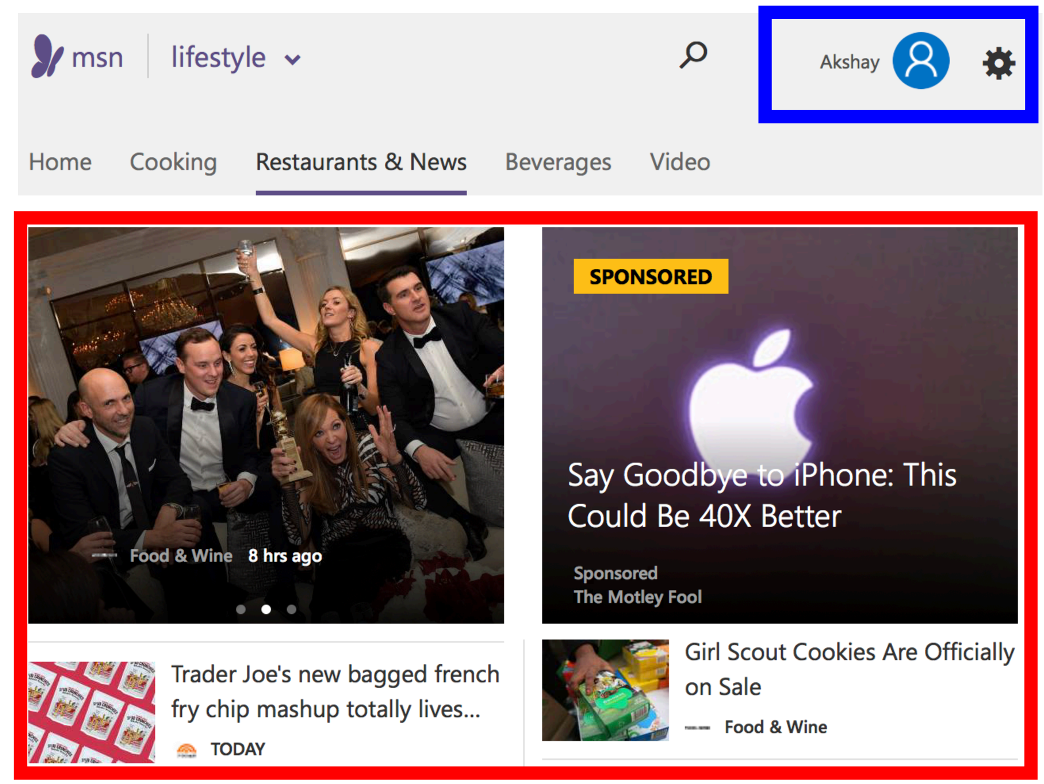


Combinatorial Contextual Bandits

On each of T rounds:

1. Observe context x_t
2. Play action $A_t = (a_1, \ldots, a_L)$
3. *Unobserved* features $\{y(\ell, a_\ell)\}_{\ell=1}^{L}$
4. Observe reward

$$r_t(A_t, x_t) = \sum_{\ell} y(\ell, a_\ell) + \text{noise}$$



B = number of simple actions
L = composite action length

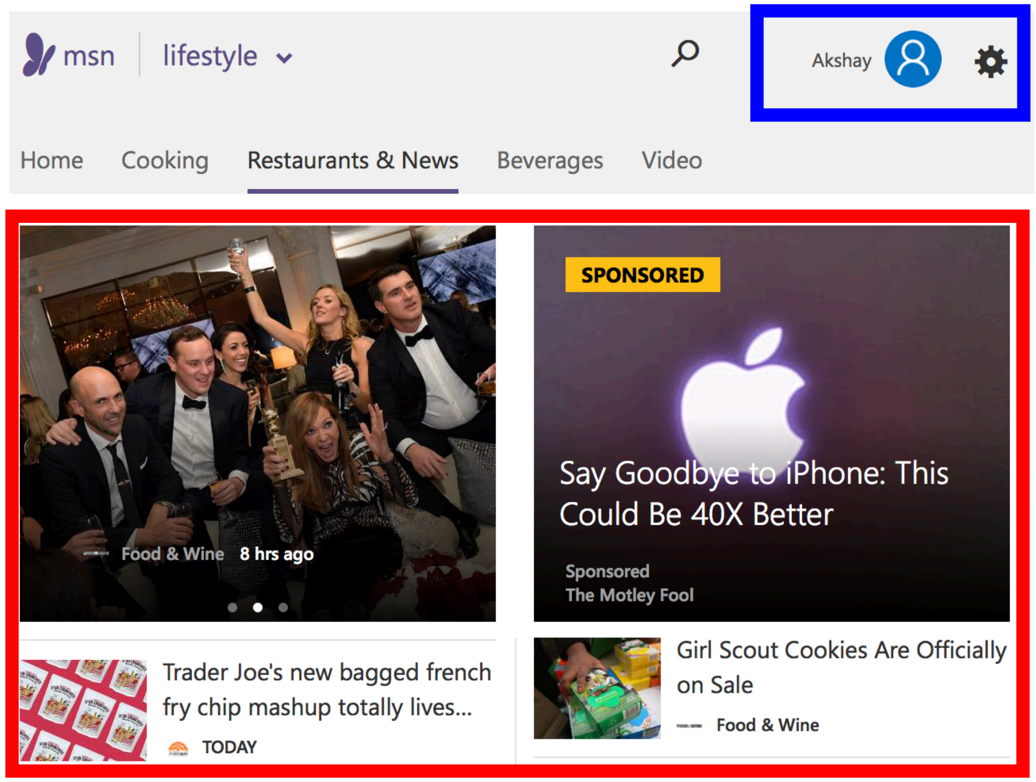
Question: Improve performance by leveraging reward structure?

Combinatorial Contextual Bandits

On each of T rounds:

1. Observe context x_t
2. Play action $A_t = (a_1, \ldots, a_L)$
3. *Unobserved* features $\{y(\ell, a_\ell)\}_{\ell=1}^{L}$
4. Observe reward

$$r_t(A_t, x_t) = \sum_{\ell} y(\ell, a_\ell) + \text{noise}$$



B = number of simple actions
L = composite action length

Question: Improve performance by leveraging reward structure?

Challenges:

- Off-policy evaluation?
- Explore vs Exploit?
- Computational Efficiency?

Results — Off-policy evaluation

Subproblem: Given data collected by logging policy, estimate reward of a target policy

[Swaminathan, Krishnamurthy, Agarwal, Dudik, Langford, Zitouni, Jose. NeurIPS 2017]

Results — Off-policy evaluation

Subproblem: Given data collected by logging policy, estimate reward of a target policy

Theorem: If logging close to uniform, can estimate target with BL/ϵ^2 samples

Parameters: B simple actions, composite action length L

[Swaminathan, Krishnamurthy, Agarwal, Dudik, Langford, Zitouni, Jose. NeurIPS 2017]

Results — Off-policy evaluation

Subproblem: Given data collected by logging policy, estimate reward of a target policy

Theorem: If logging close to uniform, can estimate target with BL/ϵ^2 samples

Parameters: B simple actions, composite action length L

- Compare with $O(B^L)$ naively and $O(B)$ with semibandit feedback

[Swaminathan, Krishnamurthy, Agarwal, Dudik, Langford, Zitouni, Jose. NeurIPS 2017]

Results — Off-policy evaluation

Subproblem: Given data collected by logging policy, estimate reward of a target policy

Theorem: If logging close to uniform, can estimate target with BL/ϵ^2 samples

Parameters: B simple actions, composite action length L

- Compare with $O(B^L)$ naively and $O(B)$ with semibandit feedback
- Gives decent eps-greedy algorithm with $T^{2/3}(BL)^{1/3}$ regret

[Swaminathan, Krishnamurthy, Agarwal, Dudik, Langford, Zitouni, Jose. NeurIPS 2017]

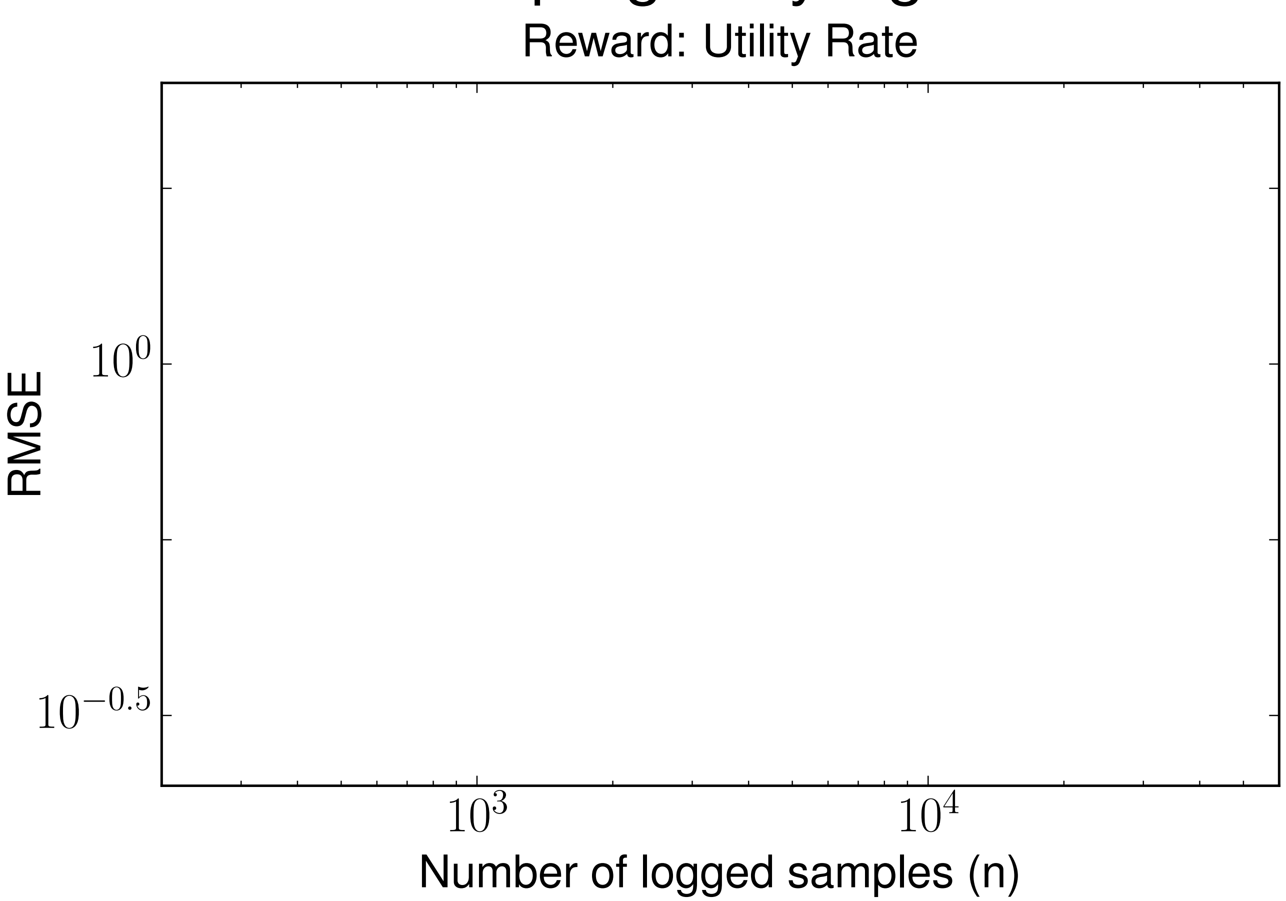
Results — Off-policy evaluation

Subproblem: Given data collected by logging policy, estimate reward of a target policy

Theorem: If logging close to uniform, can estimate target with BL/ϵ^2 samples

Parameters: B simple actions, composite action length L

- Compare with $O(B^L)$ naively and $O(B)$ with semibandit feedback
- Gives decent eps-greedy algorithm with $T^{2/3}(BL)^{1/3}$ regret



[Swaminathan, Krishnamurthy, Agarwal, Dudik, Langford, Zitouni, Jose. NeurIPS 2017]

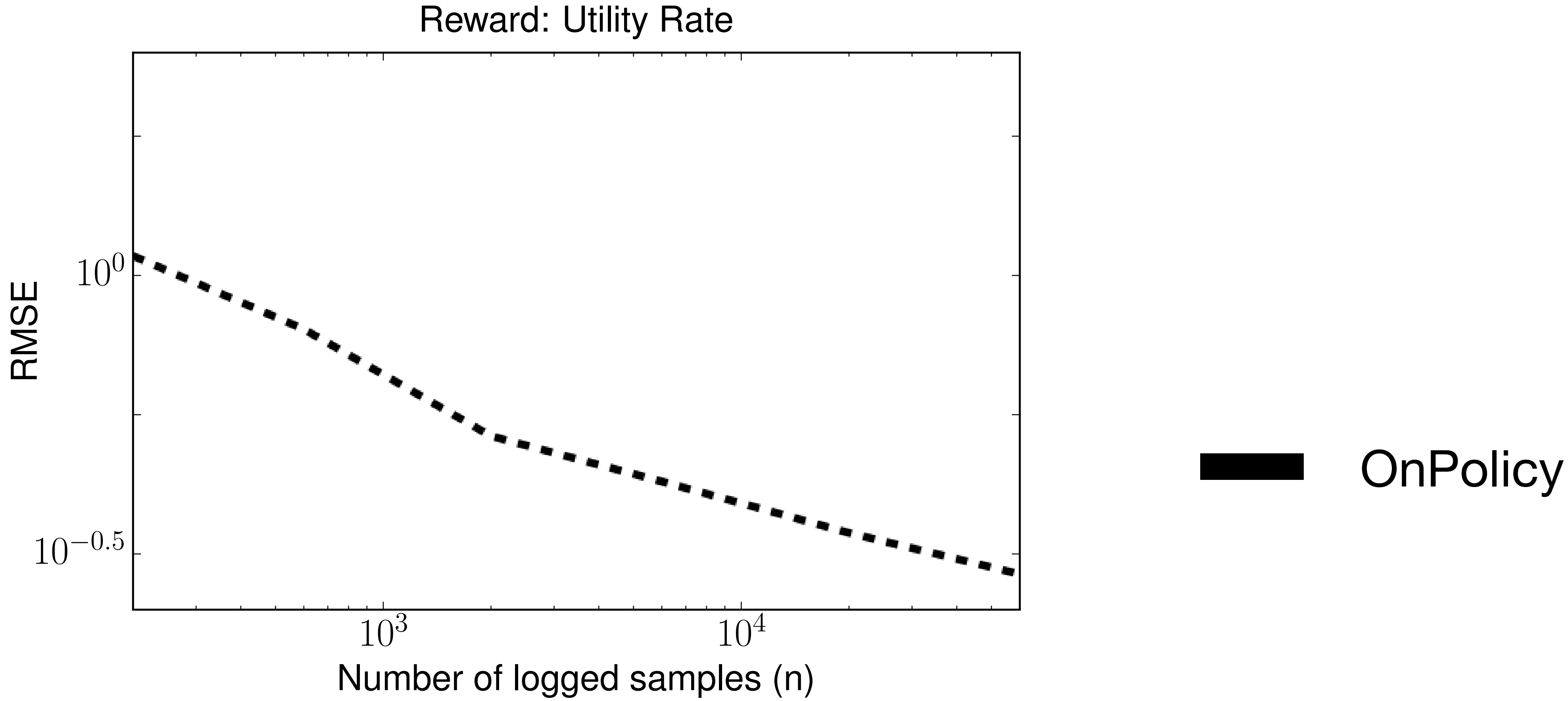
Results — Off-policy evaluation

Subproblem: Given data collected by logging policy, estimate reward of a target policy

Theorem: If logging close to uniform, can estimate target with BL/ϵ^2 samples

Parameters: B simple actions, composite action length L

- Compare with $O(B^L)$ naively and $O(B)$ with semibandit feedback
- Gives decent eps-greedy algorithm with $T^{2/3}(BL)^{1/3}$ regret



[Swaminathan, Krishnamurthy, Agarwal, Dudik, Langford, Zitouni, Jose. NeurIPS 2017]

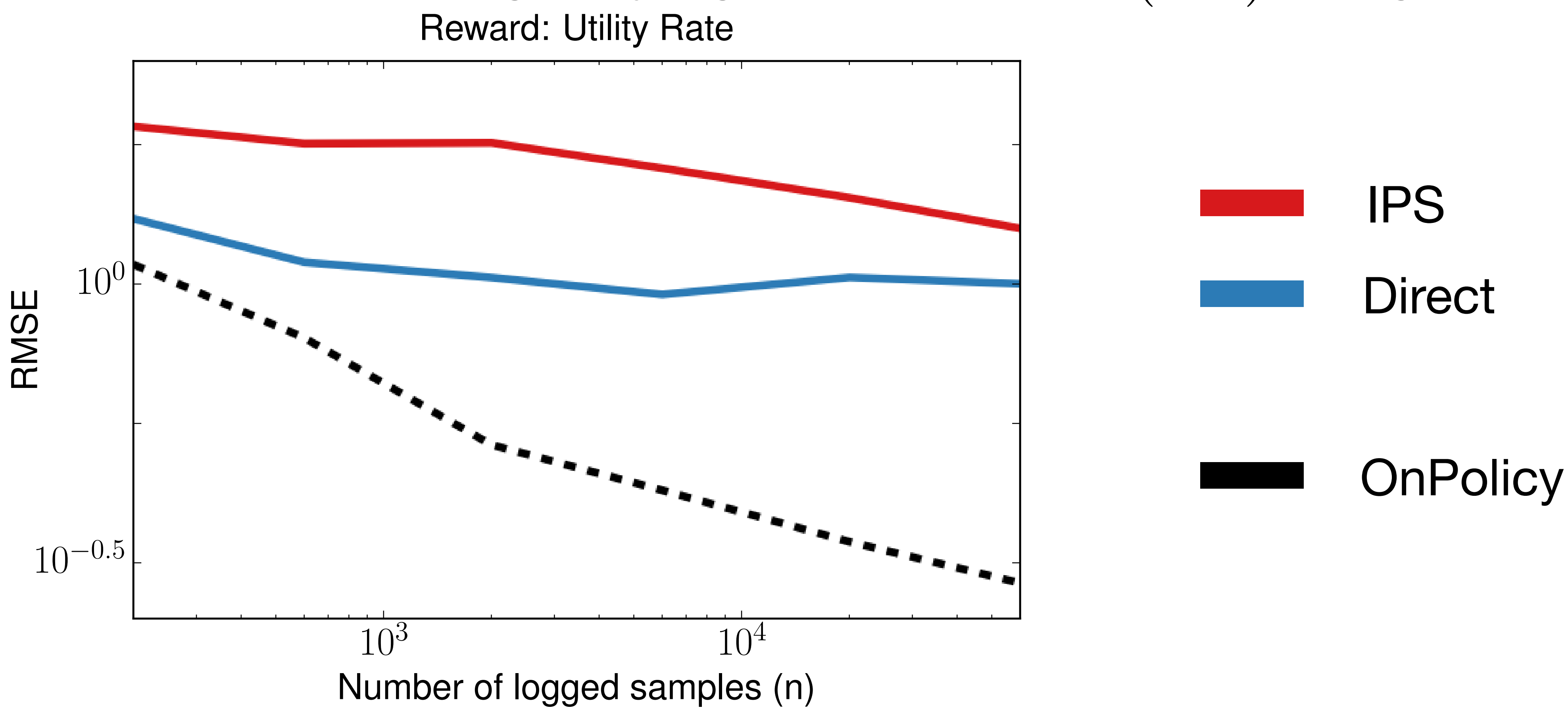
Results — Off-policy evaluation

Subproblem: Given data collected by logging policy, estimate reward of a target policy

Theorem: If logging close to uniform, can estimate target with BL/ϵ^2 samples

Parameters: B simple actions, composite action length L

- Compare with $O(B^L)$ naively and $O(B)$ with semibandit feedback
- Gives decent eps-greedy algorithm with $T^{2/3}(BL)^{1/3}$ regret



[Swaminathan, Krishnamurthy, Agarwal, Dudik, Langford, Zitouni, Jose. NeurIPS 2017]

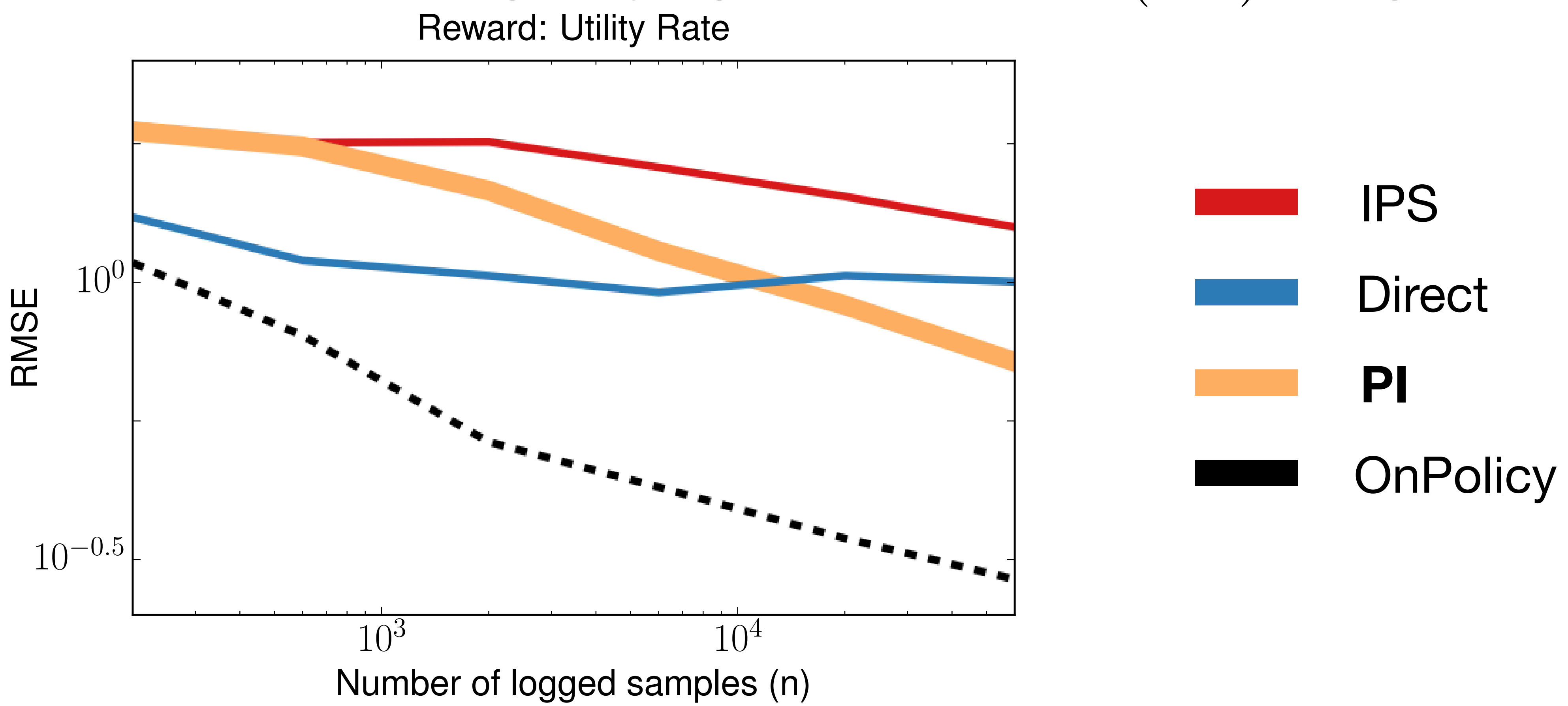
Results — Off-policy evaluation

Subproblem: Given data collected by logging policy, estimate reward of a target policy

Theorem: If logging close to uniform, can estimate target with BL/ϵ^2 samples

Parameters: B simple actions, composite action length L

- Compare with $O(B^L)$ naively and $O(B)$ with semibandit feedback
- Gives decent eps-greedy algorithm with $T^{2/3}(BL)^{1/3}$ regret

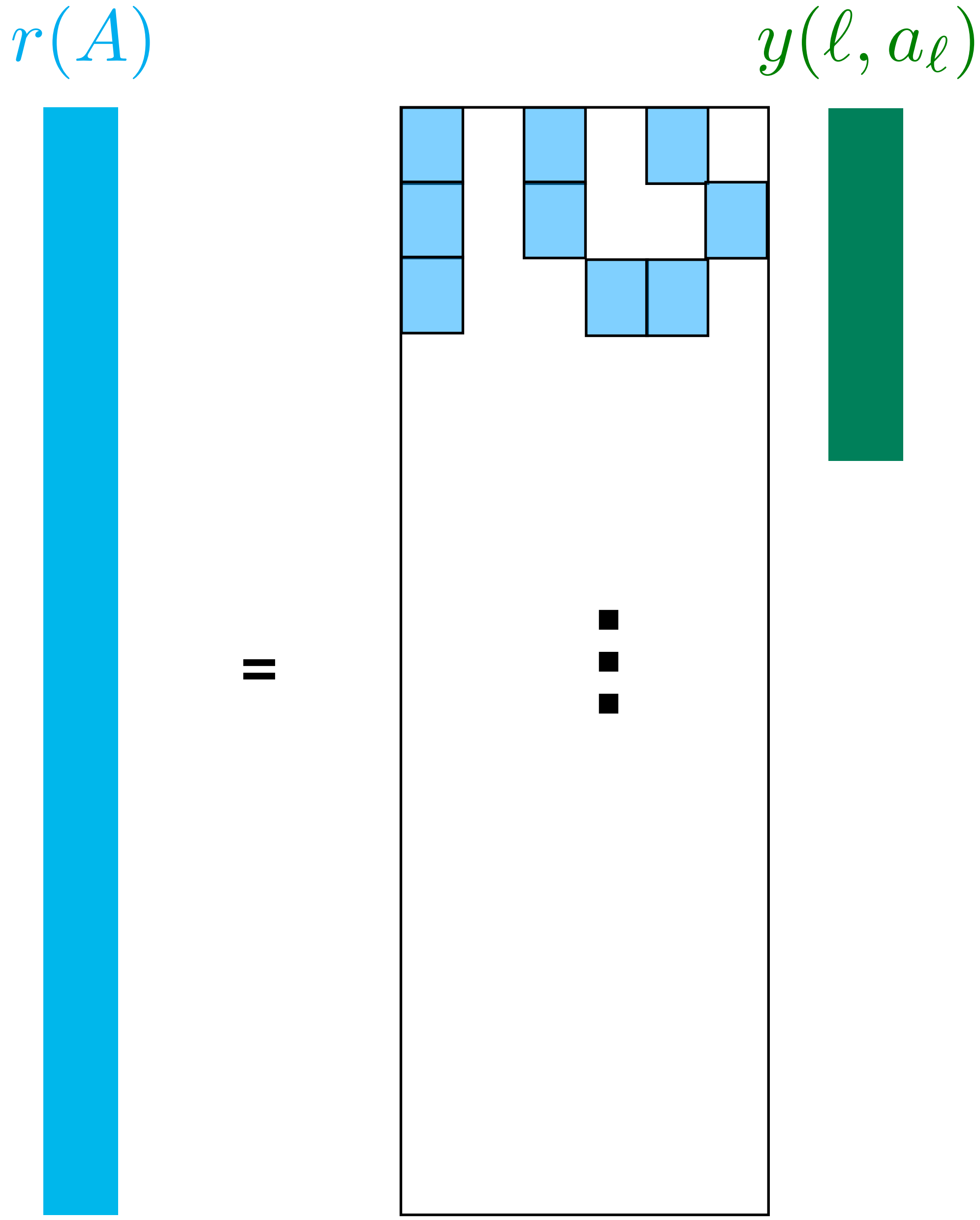


[Swaminathan, Krishnamurthy, Agarwal, Dudik, Langford, Zitouni, Jose. NeurIPS 2017]

Techniques

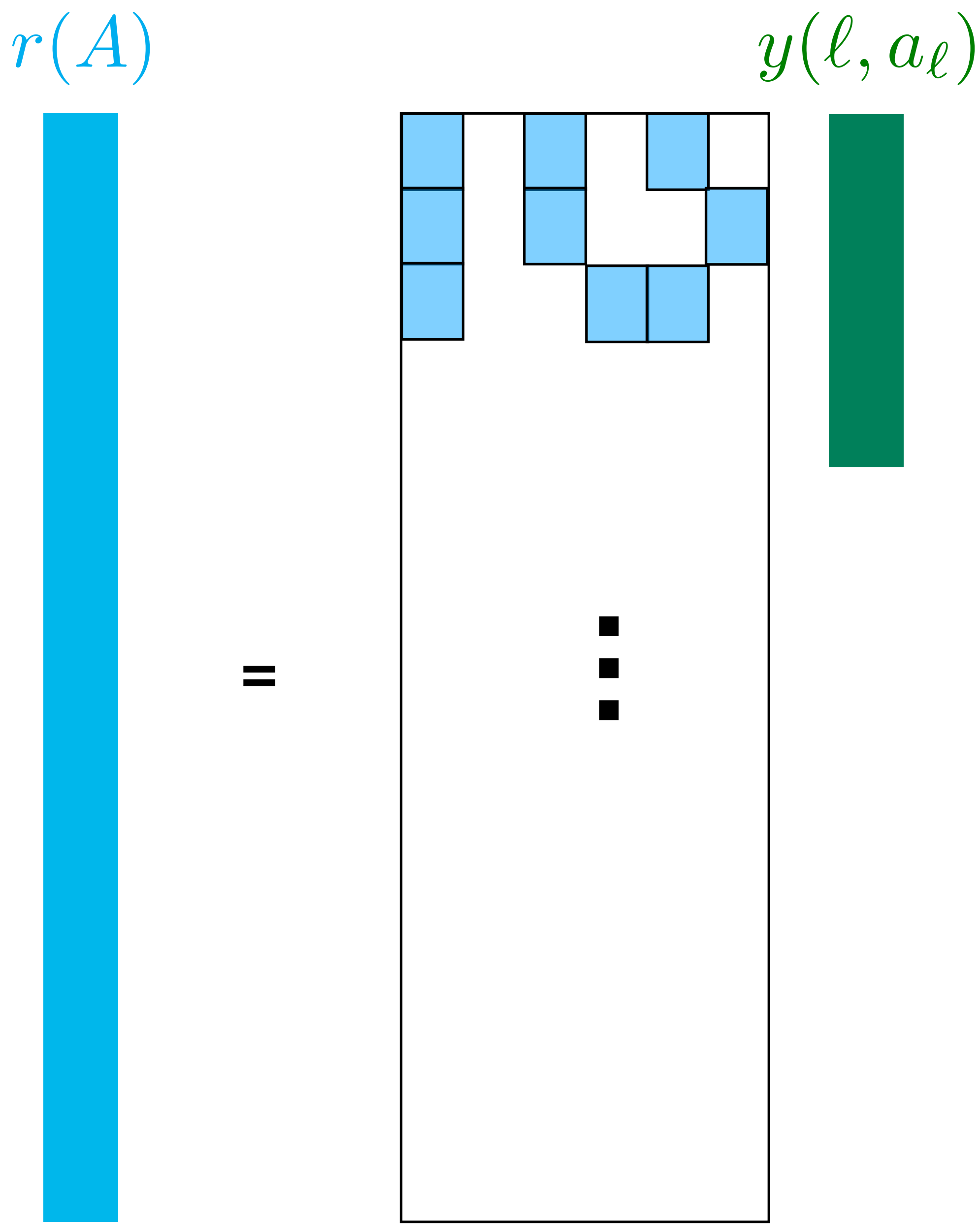
$$r_t(A_t, x_t) = \sum_{\ell} y(a_\ell) + \text{noise}$$

Techniques



$$r_t(A_t, x_t) = \sum_\ell y(a_\ell) + \text{noise}$$

Techniques

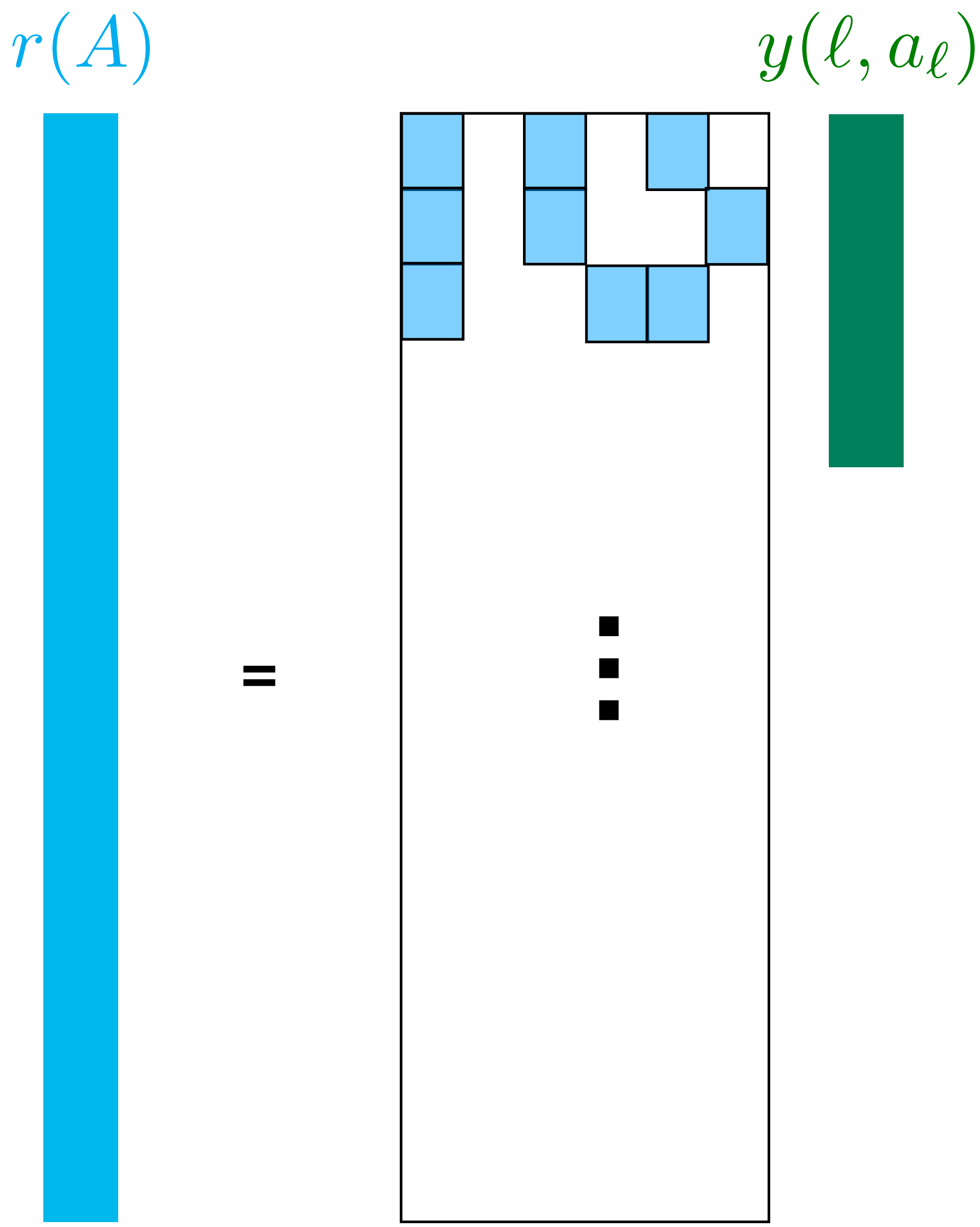


$$r_t(A_t, x_t) = \sum_{\ell} y(a_\ell) + \text{noise}$$

With logging μ can write

$$\bar{y} = \arg\min_{w} \mathbb{E}_{\mu}[(\mathbf{1}_A^T w - r)^2 | x]$$

Techniques



$$r_t(A_t, x_t) = \sum_{\ell} y(a_\ell) + \text{noise}$$

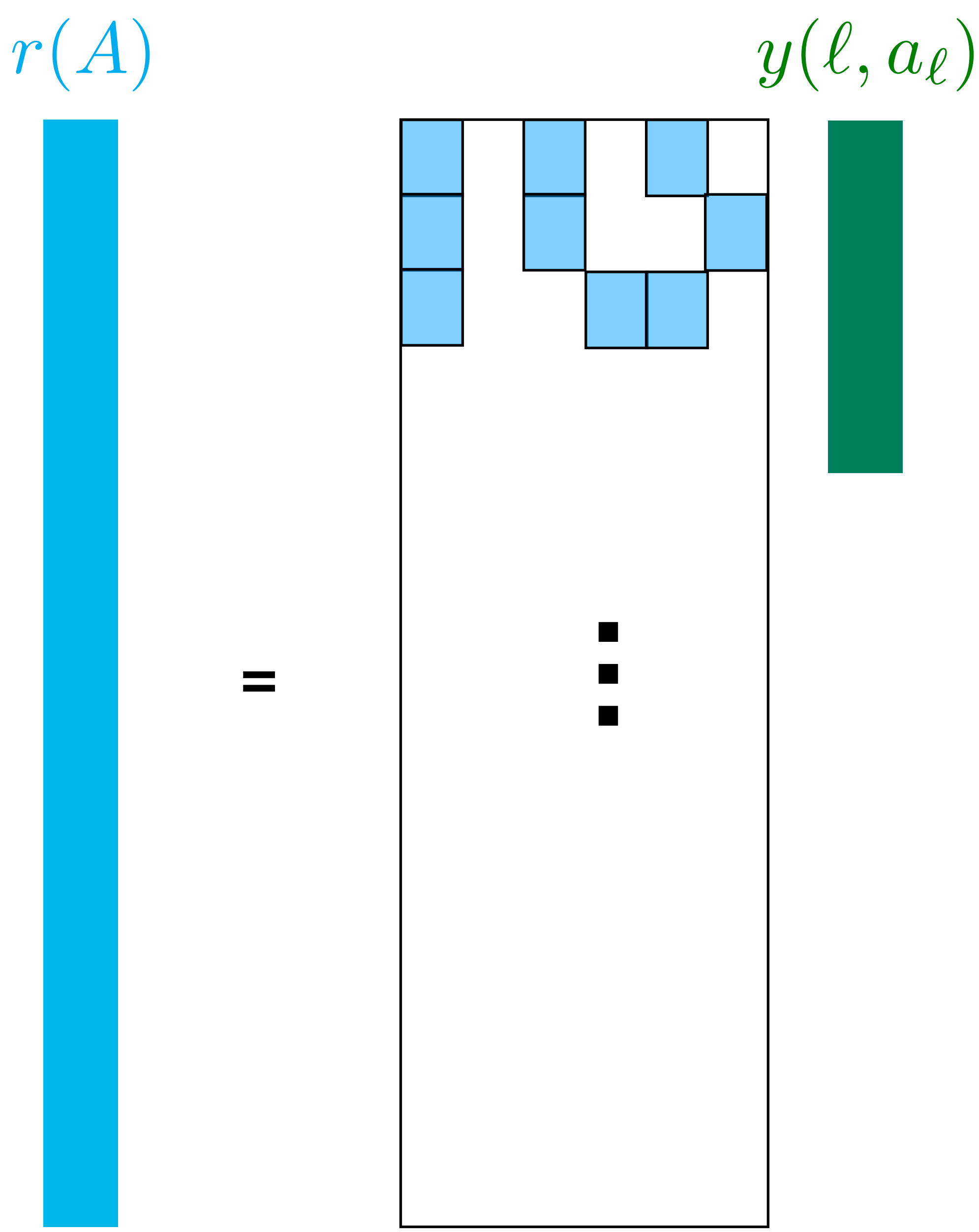
With logging μ can write

$$\bar{y} = \arg\min_{w} \mathbb{E}_\mu[(\mathbf{1}_A^T w - r)^2 | x]$$

So with (A_t, r_t) estimate

$$\hat{y}_t = (\mathbb{E}_\mu[\mathbf{1}_A \mathbf{1}_A^T])^\dagger \mathbf{1}_{A_t} r_t$$

Techniques



$$r_t(A_t, x_t) = \sum_\ell y(a_\ell) + \text{noise}$$

With logging μ can write

$$\bar{y} = \arg\min_w \mathbb{E}_\mu[(\mathbf{1}_A^T w - r)^2 | x]$$

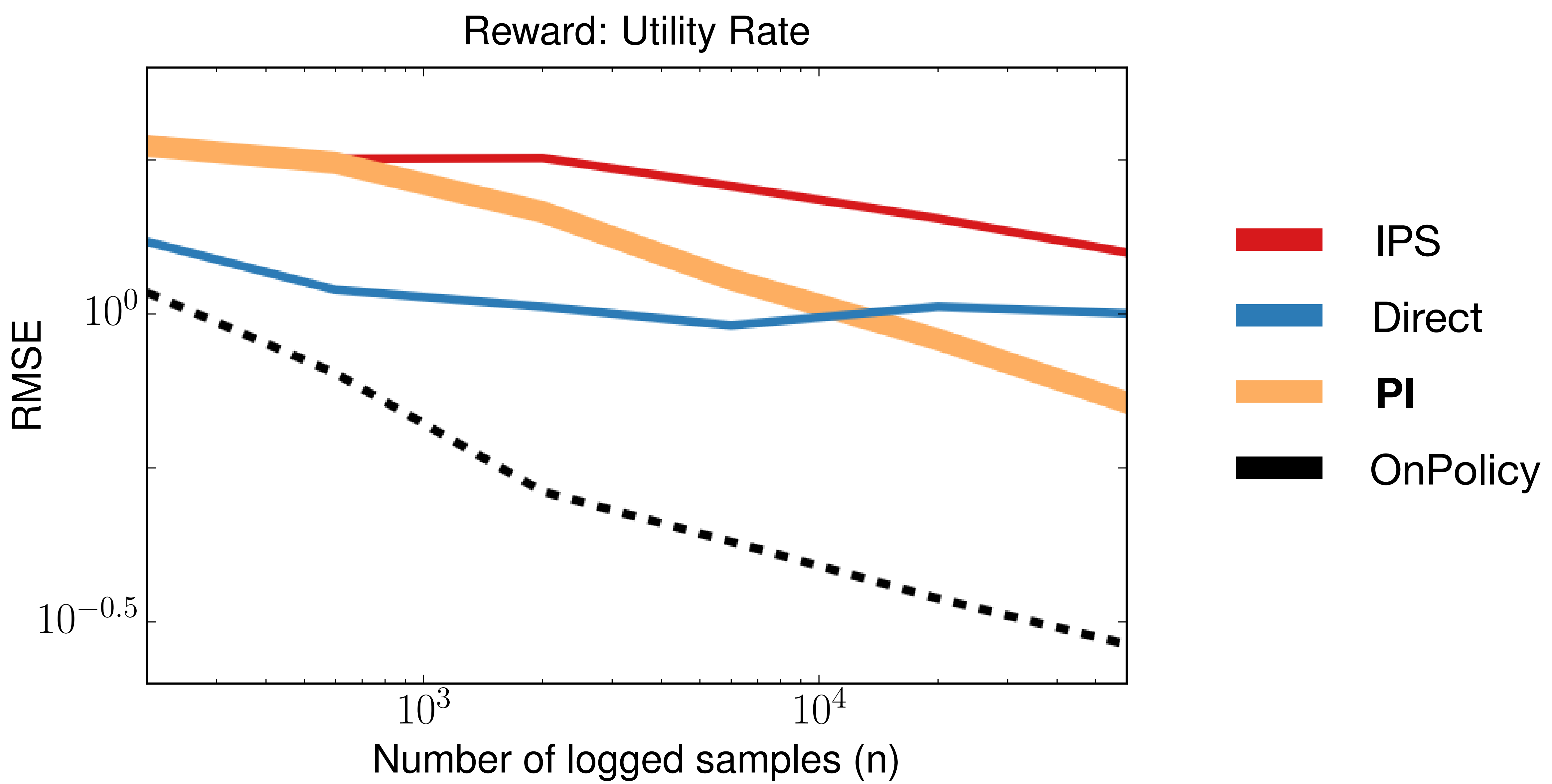
So with (A_t, r_t) estimate

$$\hat{y}_t = (\mathbb{E}_\mu[\mathbf{1}_A \mathbf{1}_A^T])^\dagger \mathbf{1}_{A_t} r_t$$

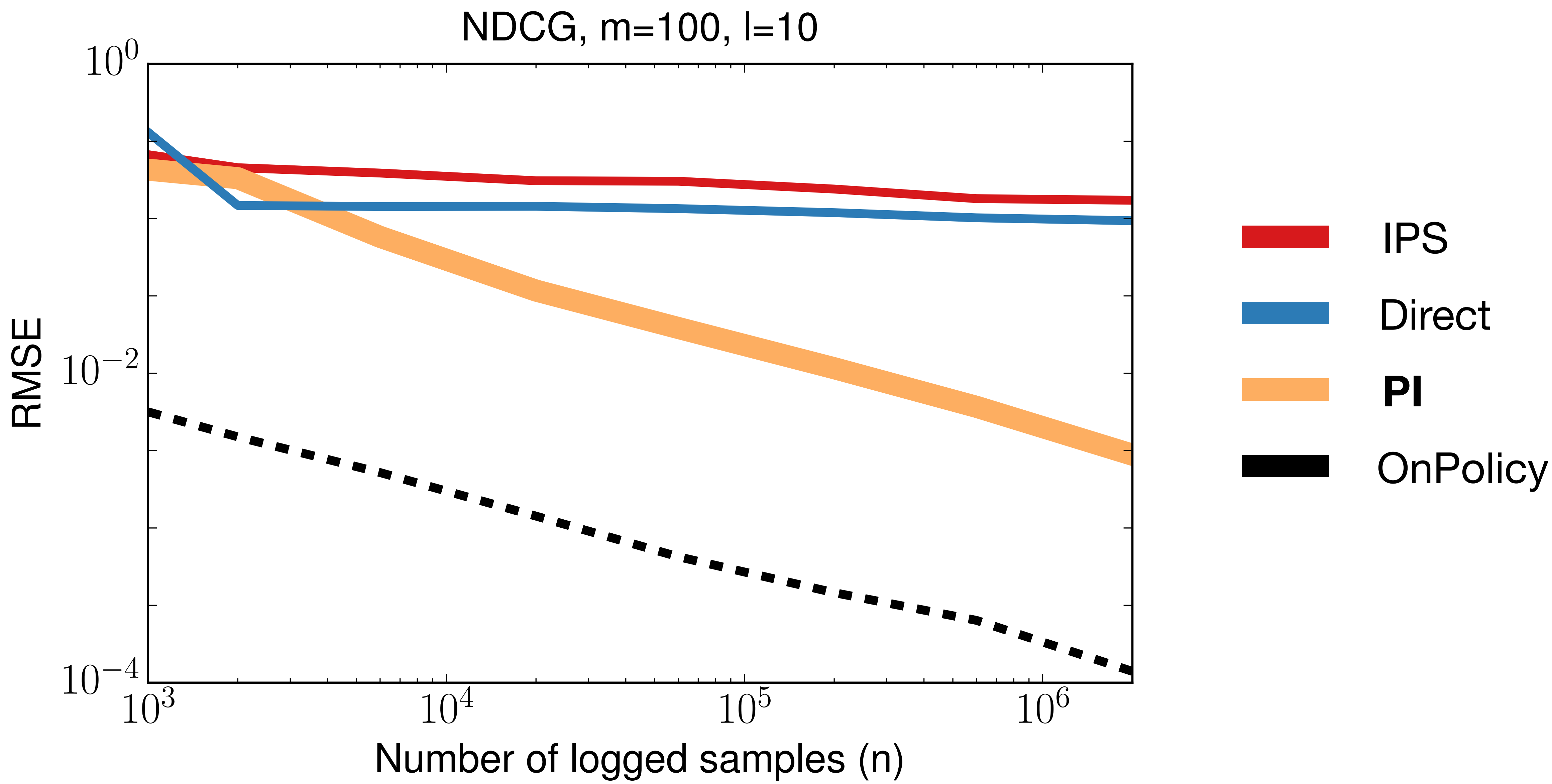
For policy π

$$\hat{r}(\pi, x_t) = \mathbf{1}_{\pi(x_t)}^T \hat{y}_t$$

Experiment


Reward: Utility Rate
RMSE
10^0
$10^{-0.5}$
10^3
10^4
Number of logged samples (n)
IPS
Direct
PI
OnPolicy

Experiment



NDCG, m=100, l=10
RMSE
10^0
10^{-2}
10^{-4}
10^3
10^4
10^5
10^6
Number of logged samples (n)
IPS
Direct
PI
OnPolicy

Policy Optimization

- Use PI estimator to obtain, with x_t

$$\hat{y}_t = (\mathbb{E}_\mu[\mathbf{1}_A \mathbf{1}_A^T])^\dagger \mathbf{1}_{A_t} r_t$$

Policy Optimization

- Use PI estimator to obtain, with x_t

$$\hat{y}_t = (\mathbb{E}_\mu[\mathbf{1}_A \mathbf{1}_A^T])^\dagger \mathbf{1}_{A_t} r_t$$

- Akin to supervised learning to rank dataset

Policy Optimization

- Use PI estimator to obtain, with x_t

$$\hat{y}_t = (\mathbb{E}_\mu[\mathbf{1}_A \mathbf{1}_A^T])^\dagger \mathbf{1}_{A_t} r_t$$

- Akin to supervised learning to rank dataset
- Train L2R model via regression

Policy Optimization

- Use PI estimator to obtain, with x_t

$$\hat{y}_t = (\mathbb{E}_\mu[\mathbf{1}_A \mathbf{1}_A^T])^\dagger \mathbf{1}_{A_t} r_t$$

- Akin to supervised learning to rank dataset
- Train L2R model via regression

Metric	LambdaMART	Random	SUP	PI

Policy Optimization

- Use PI estimator to obtain, with x_t

$$\hat{y}_t = (\mathbb{E}_\mu[\mathbf{1}_A \mathbf{1}_A^T])^\dagger \mathbf{1}_{A_t} r_t$$

- Akin to supervised learning to rank dataset
- Train L2R model via regression

Metric	LambdaMART	Random	SUP	PI
NDCG	0.457	0.152	0.438	0.421

Policy Optimization

- Use PI estimator to obtain, with x_t

$$\hat{y}_t = (\mathbb{E}_\mu[\mathbf{1}_A \mathbf{1}_A^T])^\dagger \mathbf{1}_{A_t} r_t$$

- Akin to supervised learning to rank dataset
- Train L2R model via regression

Metric	LambdaMART	Random	SUP	PI
NDCG	0.457	0.152	0.438	0.421
ERR	—	0.096	0.311	**0.321**

Policy Optimization

- Use PI estimator to obtain, with x_t

$$\hat{y}_t = (\mathbb{E}_\mu[\mathbf{1}_A \mathbf{1}_A^T])^\dagger \mathbf{1}_{A_t} r_t$$

- Akin to supervised learning to rank dataset
- Train L2R model via regression

Metric	LambdaMART	Random	SUP	PI
NDCG	0.457	0.152	0.438	0.421
ERR	—	0.096	0.311	**0.321**

PI finds good targets to optimize metric!

Summary

Summary

	Naive CB	Semibandits	Combinatorial

Parameters: T rounds, B simple actions, composite action length L

Summary

	Naive CB	Semibandits	Combinatorial
Off-Policy Eval	B^L	B	BL

Parameters: T rounds, B simple actions, composite action length L

Summary

	Naive CB	Semibandits	Combinatorial
Off-Policy Eval	B^L	B	BL
Explore/Exploit	$\sqrt{B^L T \log(\|\Pi\|)}$	$\sqrt{BT \log(\|\Pi\|)}$	$T^{2/3}(BL \log(\|\Pi\|))^{1/3}$

Parameters: T rounds, B simple actions, composite action length L

Summary

	Naive CB	Semibandits	Combinatorial
Off-Policy Eval	B^L	B	BL
Explore/Exploit	$\sqrt{B^L T \log(\|\Pi\|)}$	$\sqrt{BT \log(\|\Pi\|)}$	$T^{2/3}(BL \log(\|\Pi\|))^{1/3}$

Parameters: T rounds, B simple actions, composite action length L

Empirically

Summary

	Naive CB	Semibandits	Combinatorial
Off-Policy Eval	B^L	B	BL
Explore/Exploit	$\sqrt{B^L T \log(\lvert\Pi\rvert)}$	$\sqrt{BT \log(\lvert\Pi\rvert)}$	$T^{2/3}(BL \log(\lvert\Pi\rvert))^{1/3}$

Parameters: T rounds, B simple actions, composite action length L

Empirically

- Semibandits — With rich policy class, best performance

Summary

	Naive CB	Semibandits	Combinatorial
Off-Policy Eval	B^L	B	BL
Explore/Exploit	$\sqrt{B^L T \log(\|\Pi\|)}$	$\sqrt{BT \log(\|\Pi\|)}$	$T^{2/3}(BL \log(\|\Pi\|))^{1/3}$

Parameters: T rounds, B simple actions, composite action length L

Empirically

- Semibandits — With rich policy class, best performance
- Off-Policy Eval — Works in practice, even without linearity

Summary

	Naive CB	Semibandits	Combinatorial
Off-Policy Eval	B^L	B	BL
Explore/Exploit	$\sqrt{B^L T \log(\lvert\Pi\rvert)}$	$\sqrt{BT \log(\lvert\Pi\rvert)}$	$T^{2/3}(BL \log(\lvert\Pi\rvert))^{1/3}$

Parameters: T rounds, B simple actions, composite action length L

Empirically

- Semibandits — With rich policy class, best performance
- Off-Policy Eval — Works in practice, even without linearity
- Off-Policy Opt — Finds better targets than supervision!

Summary

	Naive CB	Semibandits	Combinatorial
Off-Policy Eval	B^L	B	BL
Explore/Exploit	$\sqrt{B^L T \log(\|\Pi\|)}$	$\sqrt{BT \log(\|\Pi\|)}$	$T^{2/3}(BL \log(\|\Pi\|))^{1/3}$

Parameters: T rounds, B simple actions, composite action length L

Empirically

- Semibandits — With rich policy class, best performance
- Off-Policy Eval — Works in practice, even without linearity
- Off-Policy Opt — Finds better targets than supervision!

Open

- Efficient CCB with $\sqrt{T}$ regret