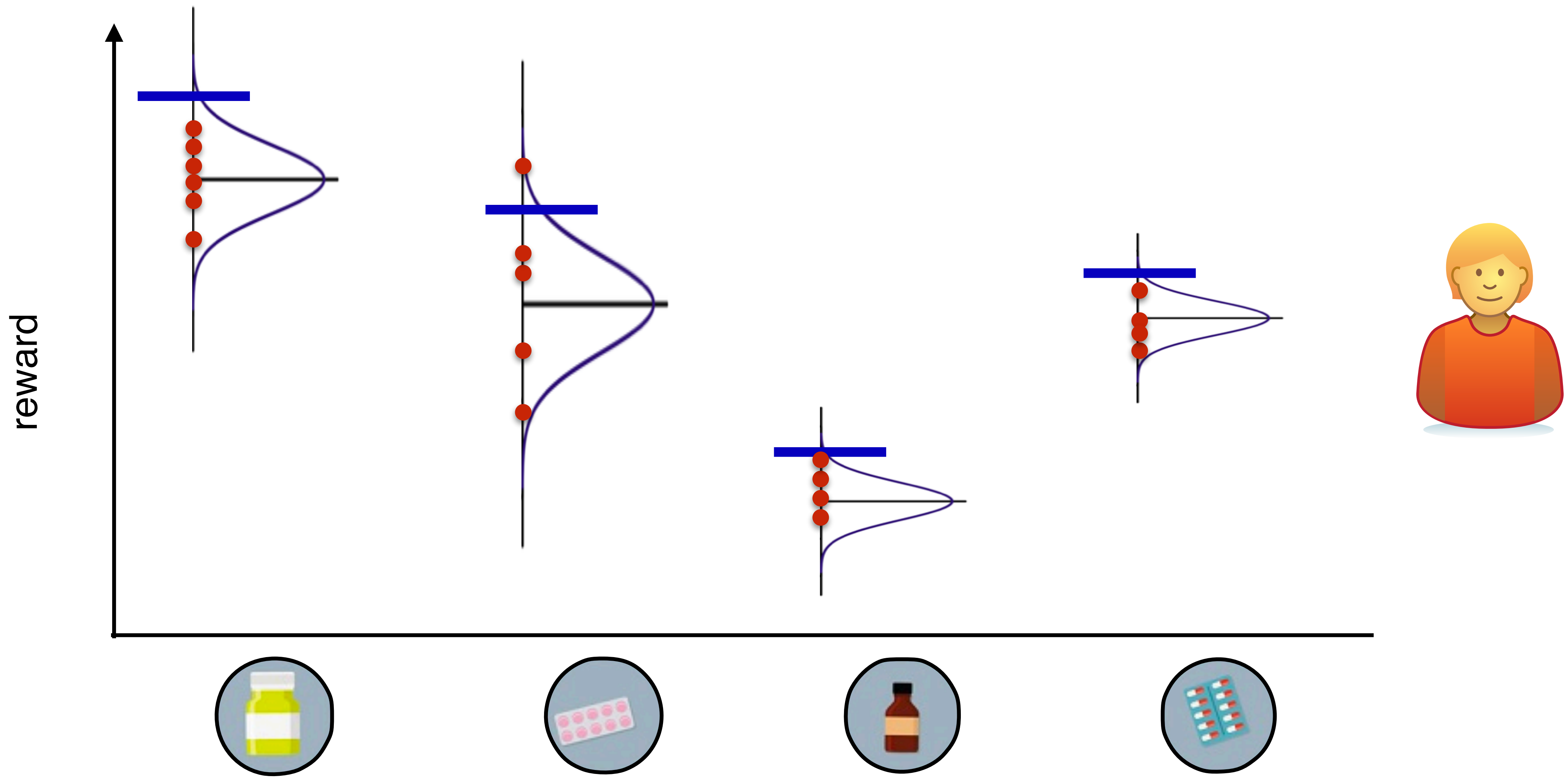
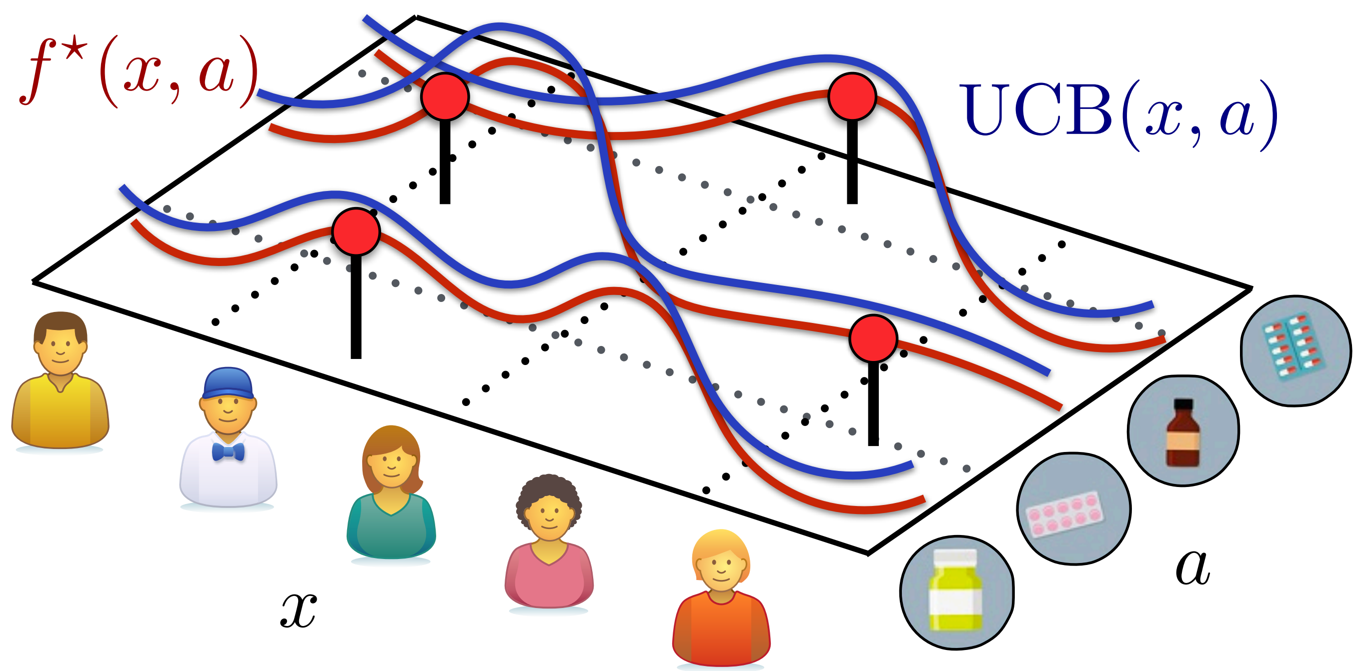
Upper confidence bound (UCB) algorithm

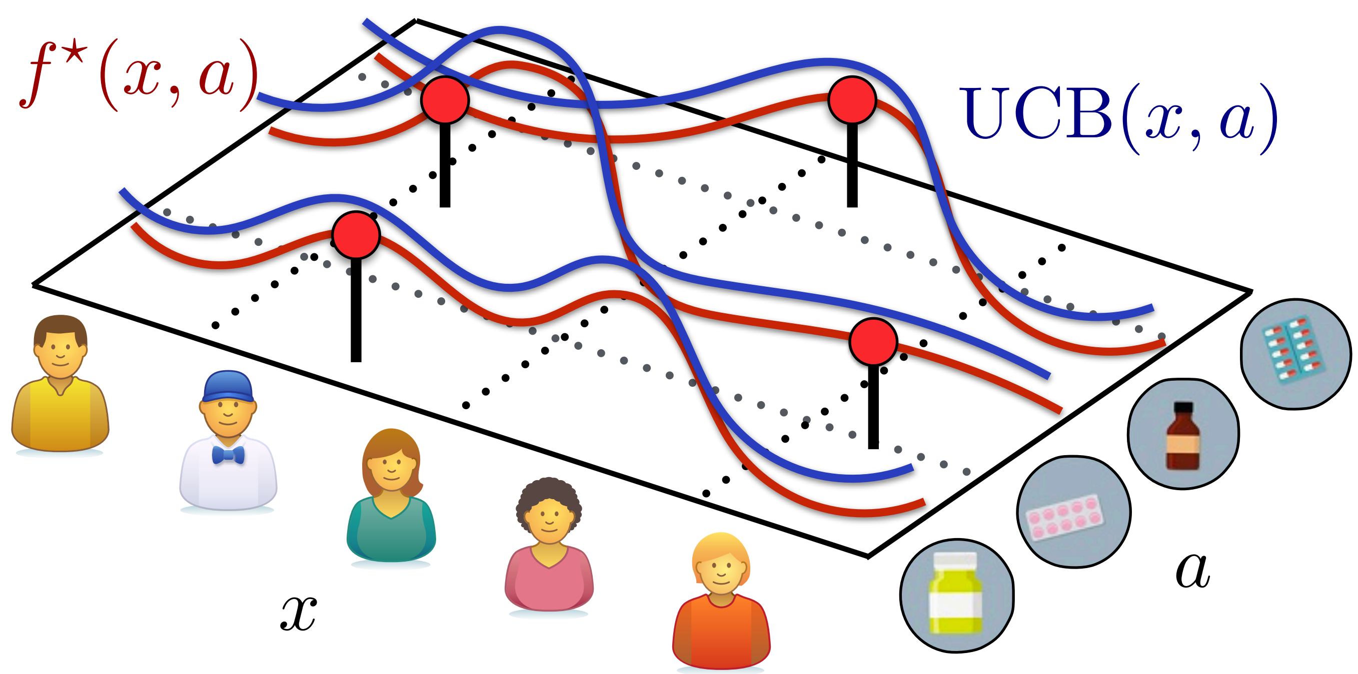


[Lai & Robbins ’85, Agrawal ’95, Auer et al. ’02]

General-purpose methods: Challenges



General-purpose methods: Challenges



- In general, no hope of constructing valid/shrinking confidence intervals for all (x, a).
- Exceptions:
  - Linear, generalized linear models (restrictive)
  - Nonparametric models (curse of dimensionality)

Lower bounds/counterexamples: [Russo & Van Roy '13], [**F**, Rakhlin, Simchi-Levi, Xu '20]

The SquareCB algorithm [F and Rakhlin'20]

SquareCB

For $t = 1, \ldots, T$:

- Receive context x_t.

The SquareCB algorithm [F and Rakhlin'20]

SquareCB

For $t = 1, \ldots, T$:

- Receive context x_t.
- Get reward estimate $\widehat{f}_t(x, a)$ from learning algorithm.

The SquareCB algorithm [F and Rakhlin'20]

SquareCB

For $t = 1, \ldots, T$:

- Receive context x_t.
- Get reward estimate $\widehat{f}_t(x, a)$ from learning algorithm.
- Assign probability p_a to each action based on $\widehat{f}_t(x_t, a)$.

The SquareCB algorithm [F and Rakhlin'20]

SquareCB

For $t = 1, \ldots, T$:

- Receive context x_t.
- Get reward estimate $\widehat{f}_t(x, a)$ from learning algorithm.
- Assign probability p_a to each action based on $\widehat{f}_t(x_t, a)$.
- Sample $a_t \sim p$ and learning algorithm with $(x_t, a_t, r_t(a_t))$.

The SquareCB algorithm [F and Rakhlin'20]

SquareCB

For $t = 1, \ldots, T$:

- Receive context x_t.
- Get reward estimate $\widehat{f}_t(x, a)$ from learning algorithm.
- Inverse Gap Weighting (**IGW**):
- Sample $a_t \sim p$ and learning algorithm with $(x_t, a_t, r_t(a_t))$.

The SquareCB algorithm [F and Rakhlin'20]

SquareCB

For $t = 1, \ldots, T$:

- Receive context x_t.
- Get reward estimate $\widehat{f}_t(x, a)$ from learning algorithm.
- Inverse Gap Weighting (**IGW**): Let $b = \arg\max_a \widehat{f}_t(x_t, a)$.
- Sample $a_t \sim p$ and learning algorithm with $(x_t, a_t, r_t(a_t))$.

The SquareCB algorithm [F and Rakhlin'20]

SquareCB

For $t = 1, \ldots, T$:

- Receive context x_t.
- Get reward estimate $\widehat{f}_t(x, a)$ from learning algorithm.
- Inverse Gap Weighting (**IGW**): Let $b = \arg\max_a \widehat{f}_t(x_t, a)$.

$$p_a = \frac{1}{A \quad + \quad \gamma \quad \times \quad (\widehat{f}_t(x_t, b) - \widehat{f}_t(x_t, a))} \quad \forall a \neq b$$

- Sample $a_t \sim p$ and learning algorithm with $(x_t, a_t, r_t(a_t))$.

The SquareCB algorithm [F and Rakhlin'20]

SquareCB

For $t = 1, \ldots, T$:

- Receive context x_t.
- Get reward estimate $\widehat{f}_t(x, a)$ from learning algorithm.
- Inverse Gap Weighting (**IGW**): Let $b = \arg\max_a \widehat{f}_t(x_t, a)$.

$$p_a = \frac{1}{A + \gamma \times \underbrace{(\widehat{f}_t(x_t, b) - \widehat{f}_t(x_t, a))}_{\text{reward gap between } b \text{ and } a}} \quad \forall a \neq b$$

- Sample $a_t \sim p$ and learning algorithm with $(x_t, a_t, r_t(a_t))$.

The SquareCB algorithm [F and Rakhlin'20]

SquareCB

For $t = 1, \ldots, T$:

- Receive context x_t.
- Get reward estimate $\widehat{f}_t(x, a)$ from learning algorithm.
- Inverse Gap Weighting (**IGW**): Let $b = \arg\max_a \widehat{f}_t(x_t, a)$.

$$p_a = \frac{1}{A + \underbrace{\gamma}_{\text{learning rate}} \times \underbrace{(\widehat{f}_t(x_t, b) - \widehat{f}_t(x_t, a))}_{\text{reward gap between } b \text{ and } a}} \quad \forall a \neq b$$

- Sample $a_t \sim p$ and learning algorithm with $(x_t, a_t, r_t(a_t))$.

The SquareCB algorithm [F and Rakhlin'20]

SquareCB

For $t = 1, \ldots, T$:

- Receive context x_t.
- Get reward estimate $\widehat{f}_t(x, a)$ from learning algorithm.
- Inverse Gap Weighting (**IGW**): Let $b = \arg\max_a \widehat{f}_t(x_t, a)$.

$$p_a = \frac{1}{\underbrace{A}_{\text{\# actions}} + \underbrace{\gamma}_{\text{learning rate}} \times \underbrace{(\widehat{f}_t(x_t, b) - \widehat{f}_t(x_t, a))}_{\text{reward gap between } b \text{ and } a}} \quad \forall a \neq b$$

- Sample $a_t \sim p$ and learning algorithm with $(x_t, a_t, r_t(a_t))$.

The SquareCB algorithm [F and Rakhlin'20]

SquareCB

For $t = 1, \ldots, T$:

- Receive context x_t.
- Get reward estimate $\widehat{f}_t(x, a)$ from learning algorithm.
- Inverse Gap Weighting (**IGW**): Let $b = \arg\max_a \widehat{f}_t(x_t, a)$.

$$p_a = \frac{1}{\underbrace{A}_{\text{\# actions}} + \underbrace{\gamma}_{\text{learning rate}} \times \underbrace{(\widehat{f}_t(x_t, b) - \widehat{f}_t(x_t, a))}_{\text{reward gap between } b \text{ and } a}} \quad \forall a \neq b$$

  with $p_b =$ remaining probability.

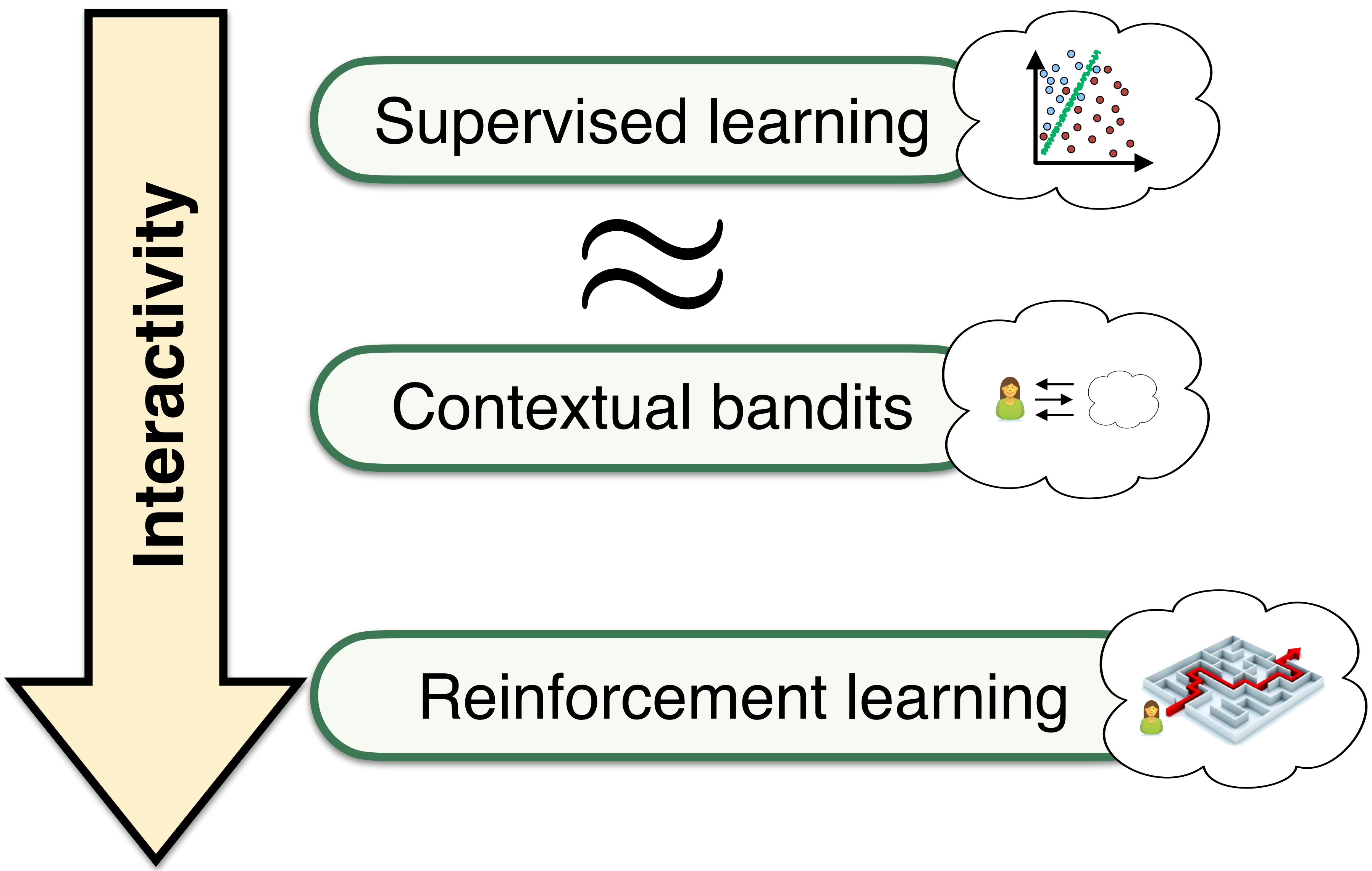
- Sample $a_t \sim p$ and learning algorithm with $(x_t, a_t, r_t(a_t))$.

SquareCB: Features

Makes decision making as easy as supervised learning!

- Use any out-of-the-box learning algorithm for $\mathcal{F}$.

  $\Longrightarrow$ **SquareCB** takes care of the rest.



Main theorem for SquareCB

Theorem [F & Rakhlin'20]

SquareCB guarantees that w.h.p.,

$$\mathbf{Reg}_{\mathsf{CB}} \leq C \cdot \sqrt{AT \cdot \mathbf{Reg}_{\mathsf{Sq}}(\mathcal{F})},$$

w/ $O(A)$ overhead in runtime and memory, where $A = \#$actions.

Optimality

More examples:

- Linear models (OLS, Ridge)
  [Abe-Long '99], [Auer '02],
  [Chu et al. '11]
- Kernels [Valko et al.'13, Zhou et al.'19]
- Generalized linear models
  [Filippi et al. '10, Li et al. '17]
- Finite classes
  [Auer et al. '01, Agarwal et al.'12]
- Sparse linear (Lasso, Elastic net)
  [Bastani & Bayati'20]
- Nonparametrics
  [Rigollet-Zeevi'10, Perchet-Rigollet'13],
  [Gur, Momeni, Wager'19]

Optimality / Universality

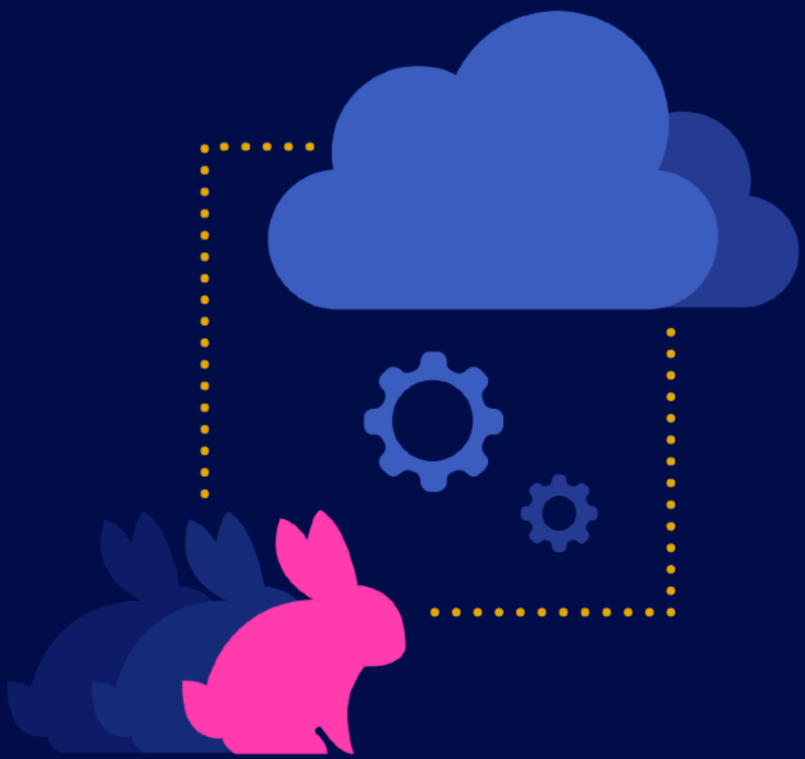
Theorem [F & Rakhlin'20]

For every function class $\mathcal{F}$, **SquareCB** attains the minimax optimal rate for CBs with $\mathcal{F}$.

To prove this, had to characterize *what* the optimal rate is [**F** & Rakhlin'20].

Vowpal Wabbit   Get started   Features   Tutorials   Research   Wiki   GitHub

Your go-to interactive machine learning library



Vowpal Wabbit provides a fast, flexible, online, and active learning solution that empowers you to solve complex interactive machine learning problems.

Get started   Tutorials

vowpalwabbit.org

What does Vowpal Wabbit do?

Vowpal Wabbit provides fast, efficient, and flexible online machine learning techniques for reinforcement learning, supervised learning, and more. It is influenced by an ecosystem of community contributions, academic research, and proven algorithms. Microsoft Research is a major contributor to Vowpal Wabbit.