Differentially Private Algorithm and Auction Configuration

Ellen Vitercik

CMU, Theory Lunch

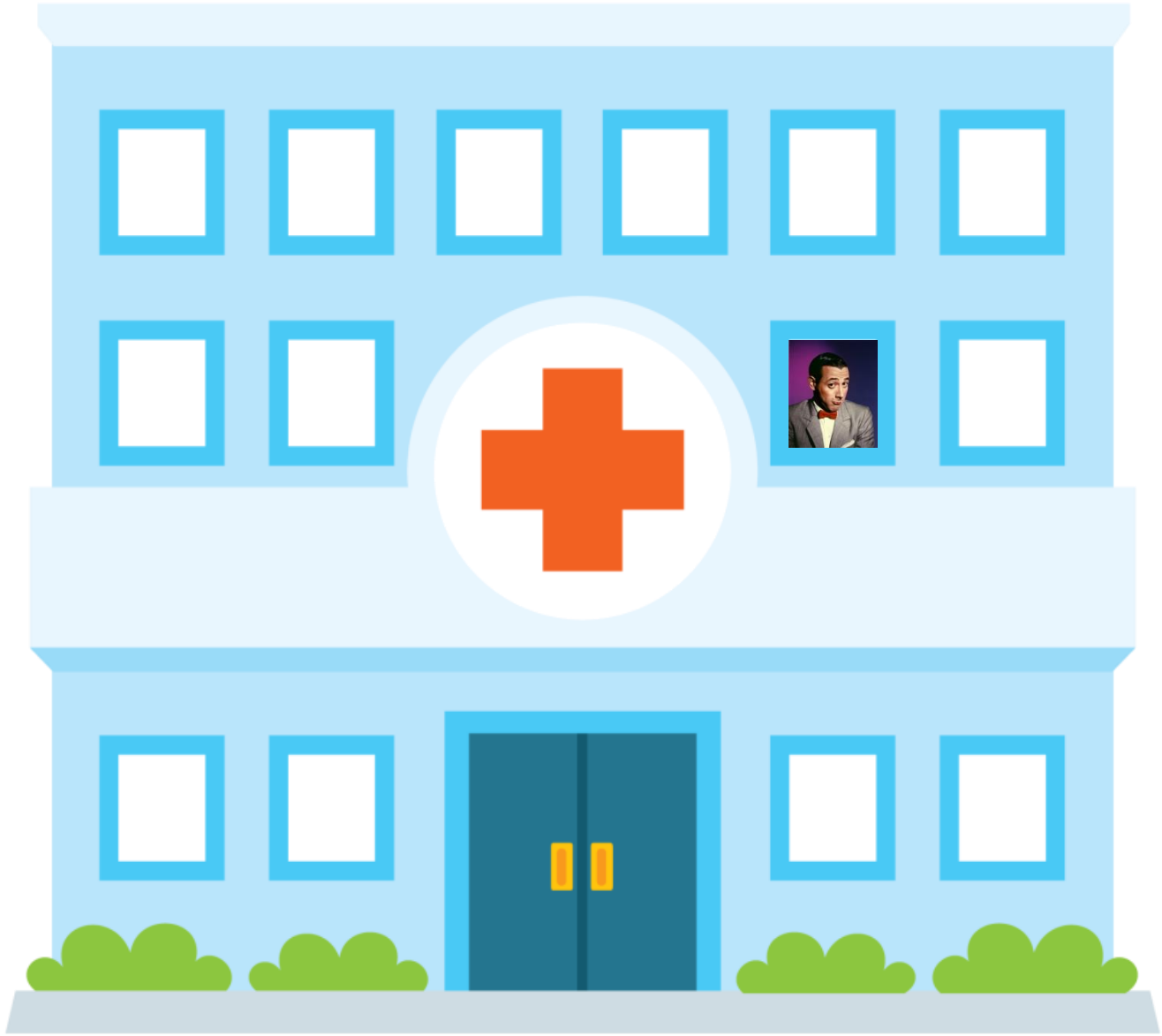
October 11, 2017

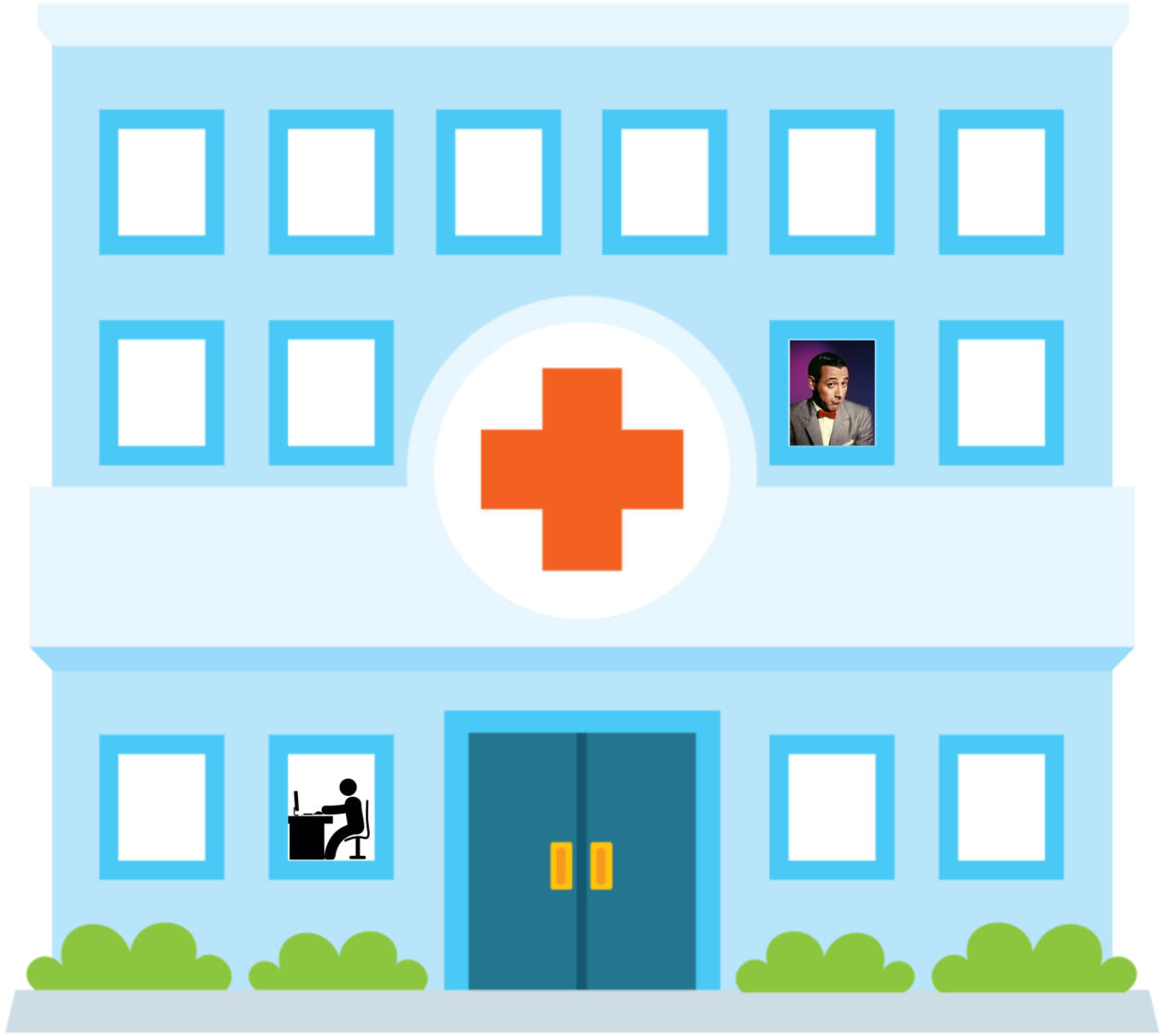
Joint work with Nina Balcan and Travis Dick


PHARMACY
$1

Prices learned from purchase histories can reveal information about individual purchases.

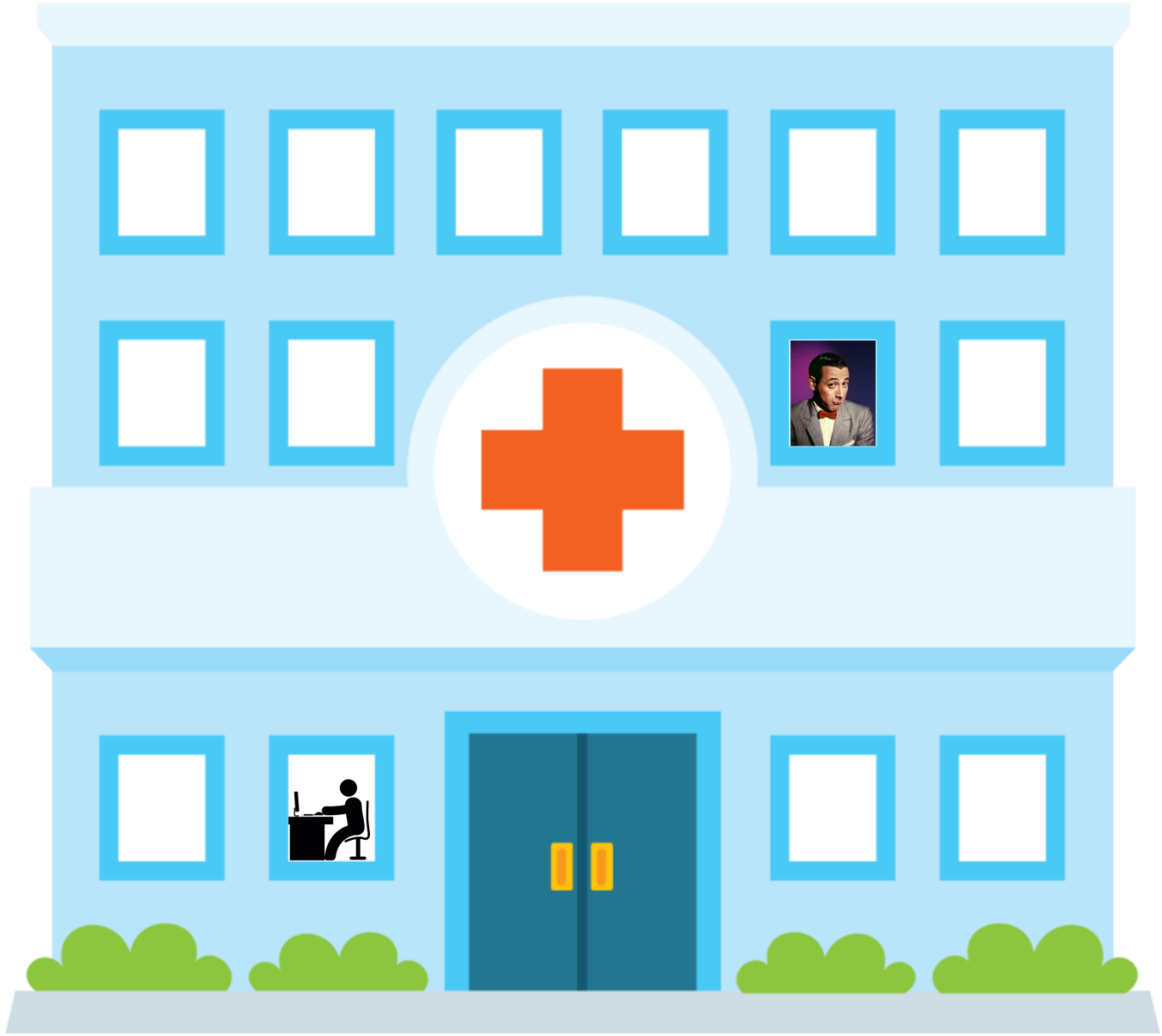
We need a way to **privately** set prices and design auctions based on purchase histories.





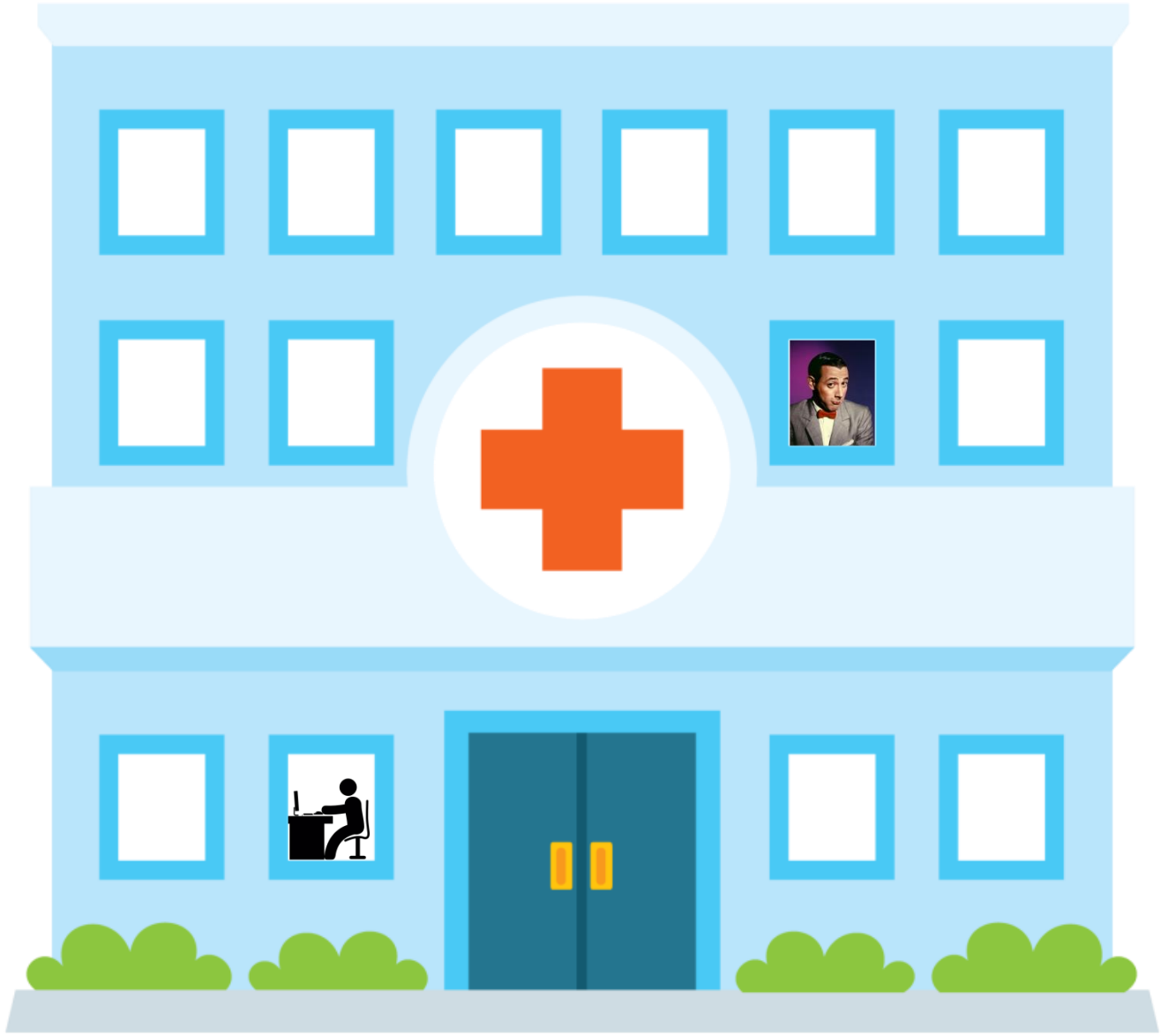
Dear Lab Technician,

Please update your default CPLEX parameters to …



Suppose a parameter correlates with a certain disease.

An attacker can infer information about medical records used to tune those parameters.



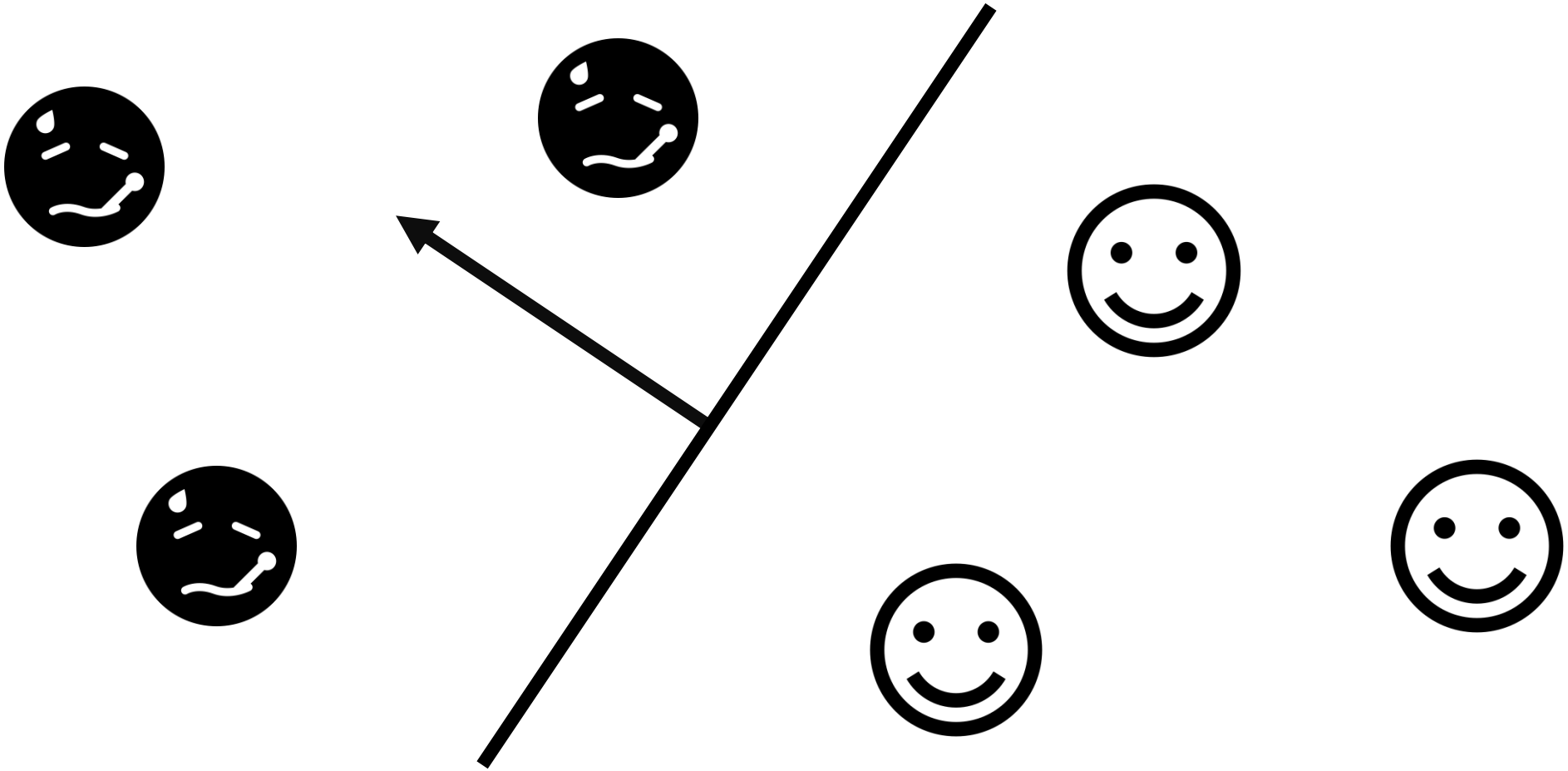
We need a way to **privately** configure algorithms.

Many works have shown that it is possible to invert a machine learning model to **infer sensitive information** about its training set.

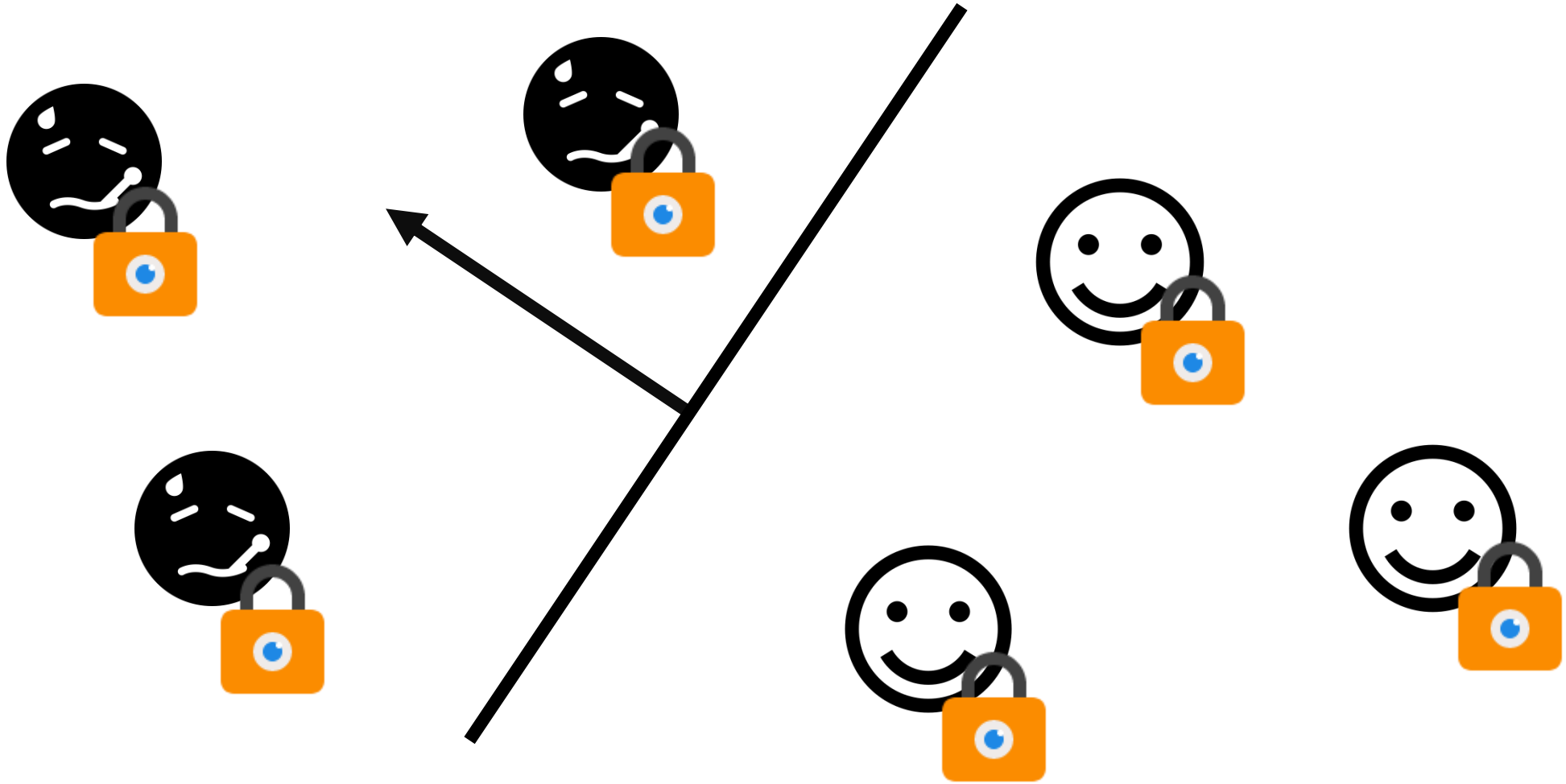
By observing a series of recommendations from websites such as Amazon, an adversary can **infer individual users' purchases.** [Calandrino et al. 2011]

It is possible to **extract images of training subjects** from facial recognition models. The attacker has only the person's name and access to a facial recognition system that returns a class confidence score. [Fredrikson et al. 2015]

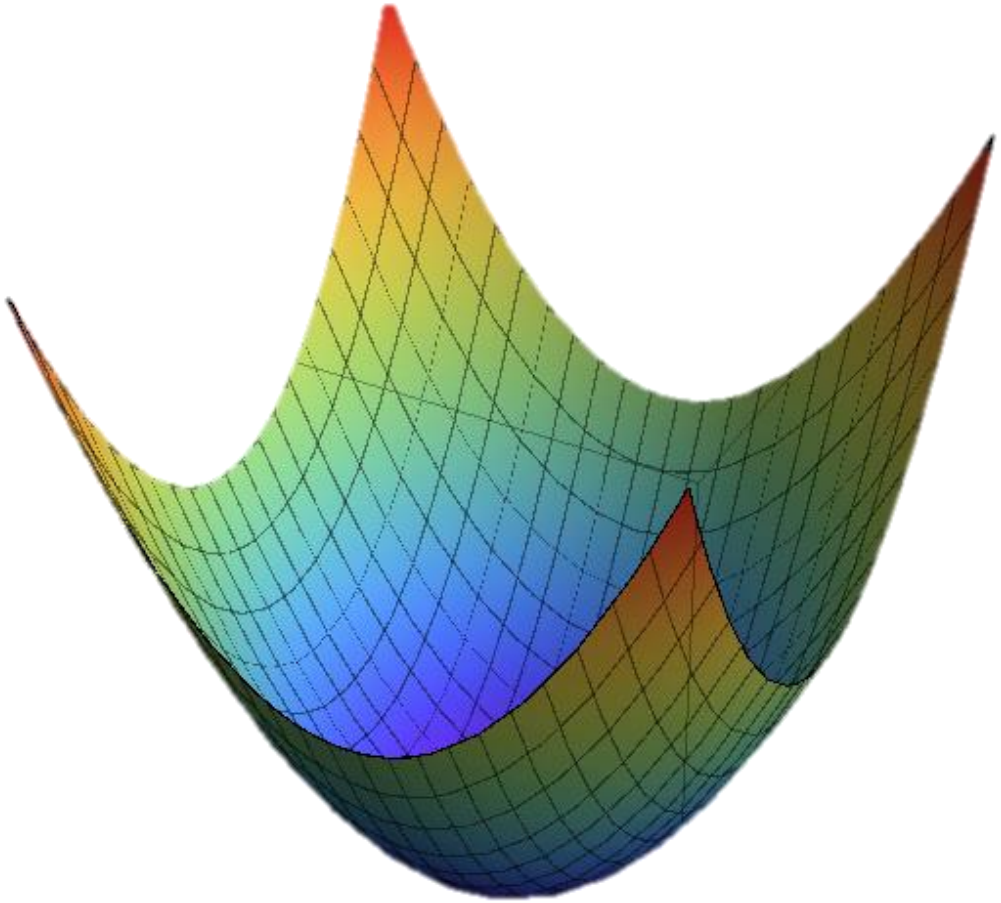
It is possible to invert a machine learning model to learn **sensitive genomic information** about individuals.
[Fredrikson et al. 2014]



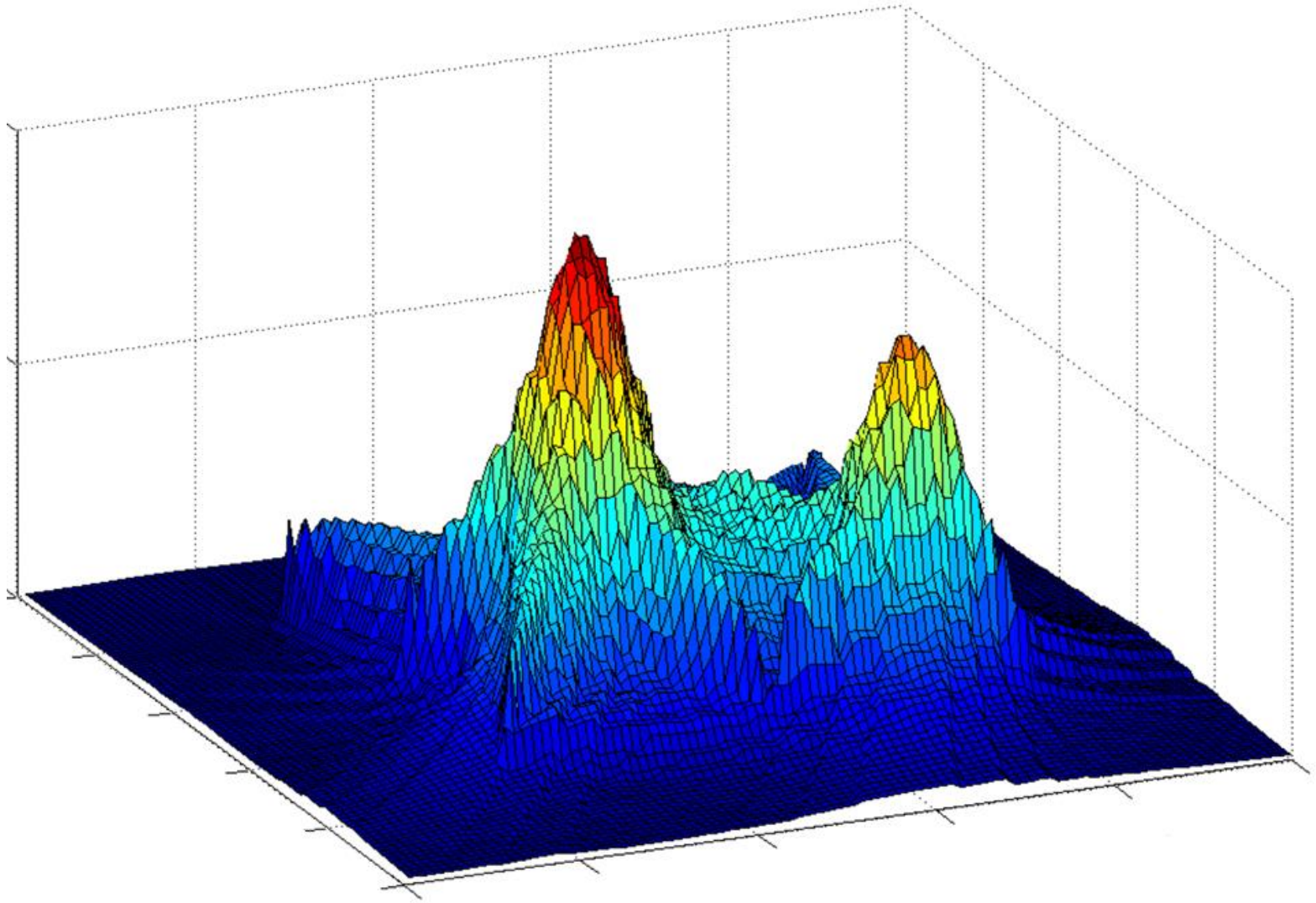
In response, computer scientists have developed private machine learning algorithms.



Existing private ML algorithms apply to optimization problems defined by well-studied, well-understood functions.

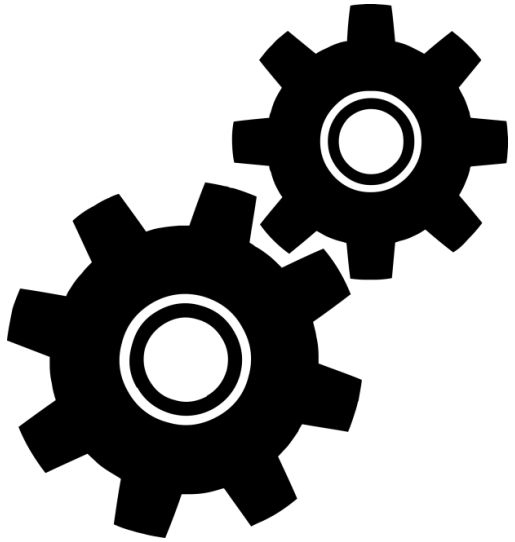


What if the objective is nonconvex and not differentiable?



We provide a **private** algorithm for maximizing data-dependent piecewise Lipschitz functions.

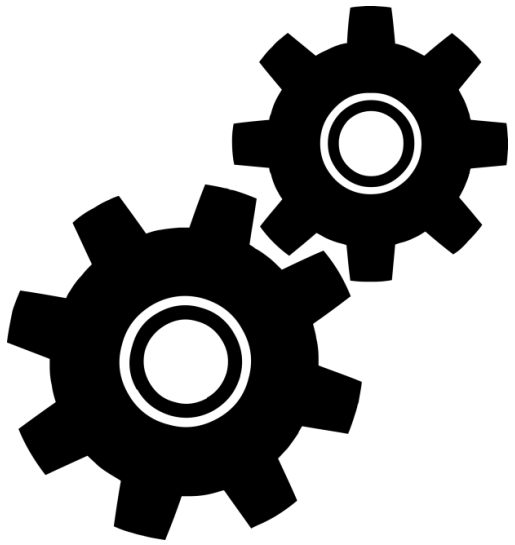
Applications in:



Algorithm configuration and mechanism design reduce to maximizing data-dependent piecewise Lipschitz functions.

- Introduction
- Setup
  - ➡ Overview: Algorithm configuration and auction design
  - Differential privacy
  - Private pricing design
- The algorithm
  - Privacy guarantees
  - Utility guarantees
    - Example: private multi-item pricing
- Summary

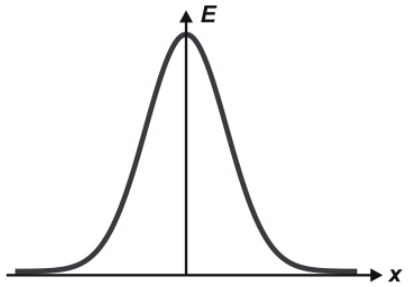
Learning-based algorithm configuration:
Tune algorithm parameters to achieve high performance over a specific application domain.



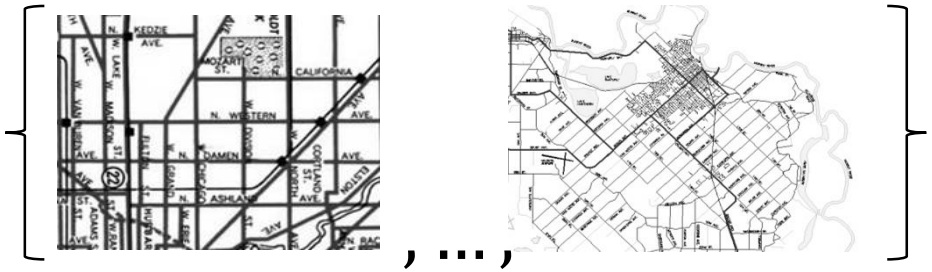
Led to breakthroughs in:

- Combinatorial auctions
  [Leyton-Brown et al., 2009]
- Scientific computing
  [Demmel et al., 2005]
- Vehicle routing
  [Caseau et al., 1999]
- SAT
  [Xu et al., 2008]

Learning-based algorithm configuration



Application-Specific Distribution



, … ,

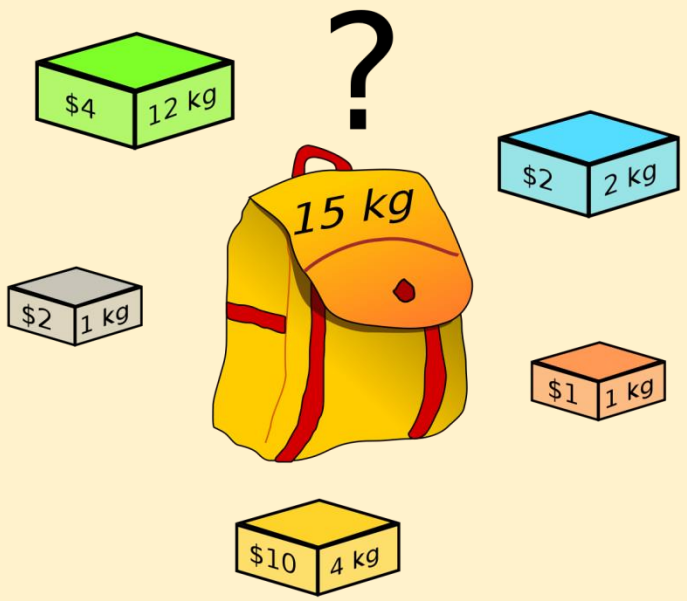
Algorithm Designer

Algorithm

How can I use the set of samples to find an algorithm that's best for my application domain?

We show that our private algorithm has strong utility guarantees for many **algorithm configuration problems**.

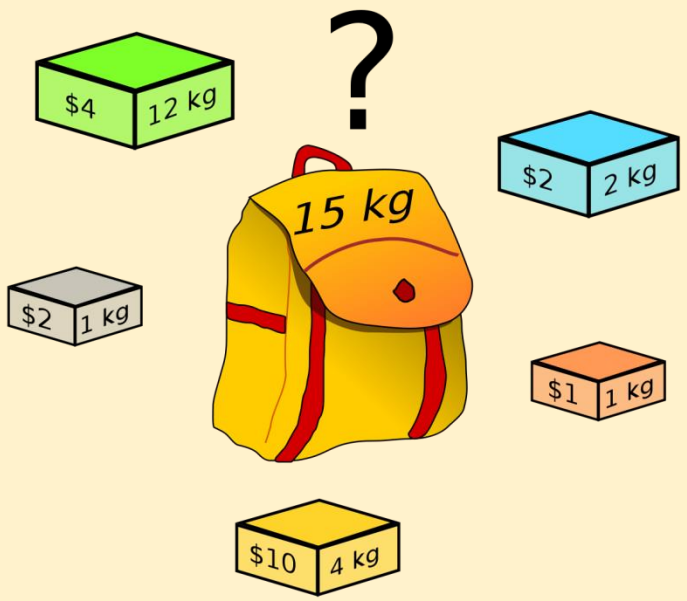
Greedy algorithm configuration



Hard combinatorial problems show up in diverse domains where **privacy preservation is crucial**.

We show that our private algorithm has strong utility guarantees for many **algorithm configuration problems**.

Greedy algorithm configuration



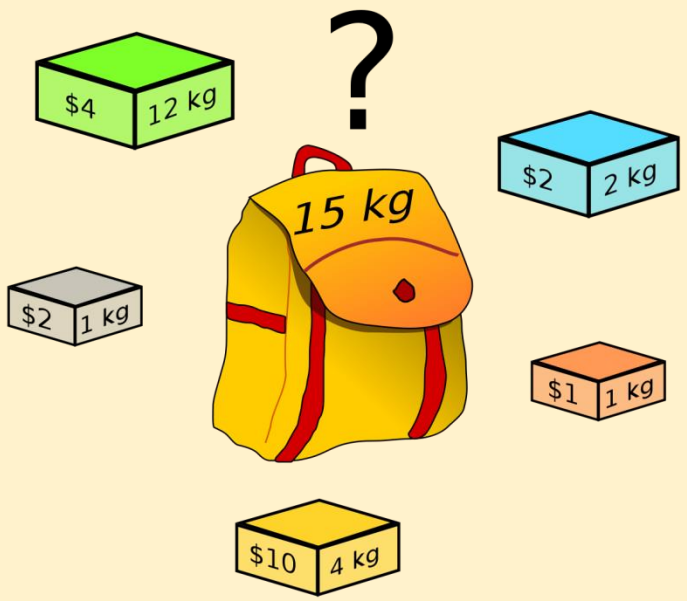
These are often solved by greedy algorithms where elements are iteratively added to a solution set according to a **heuristic**.

E.g., in knapsack:

$$\frac{(\text{size of item } i)}{(\text{value of item } i)}$$

We show that our private algorithm has strong utility guarantees for many **algorithm configuration problems**.

Greedy algorithm configuration



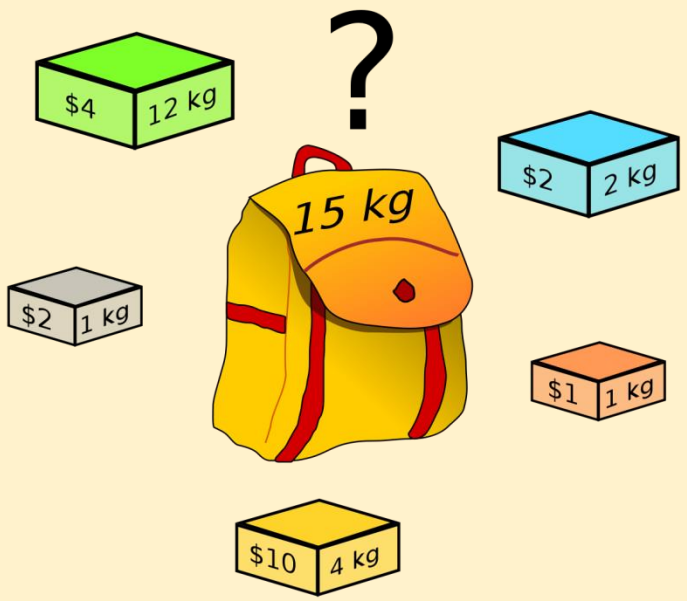
Gupta and Roughgarden [2017] proposed an **infinite family of greedy heuristics** for the knapsack and max weight independent set problems.

E.g., in knapsack:

$$\frac{(\text{size of item } i)}{(\text{value of item } i)^{\rho}}$$

We show that our private algorithm has strong utility guarantees for many **algorithm configuration problems**.

Greedy algorithm configuration

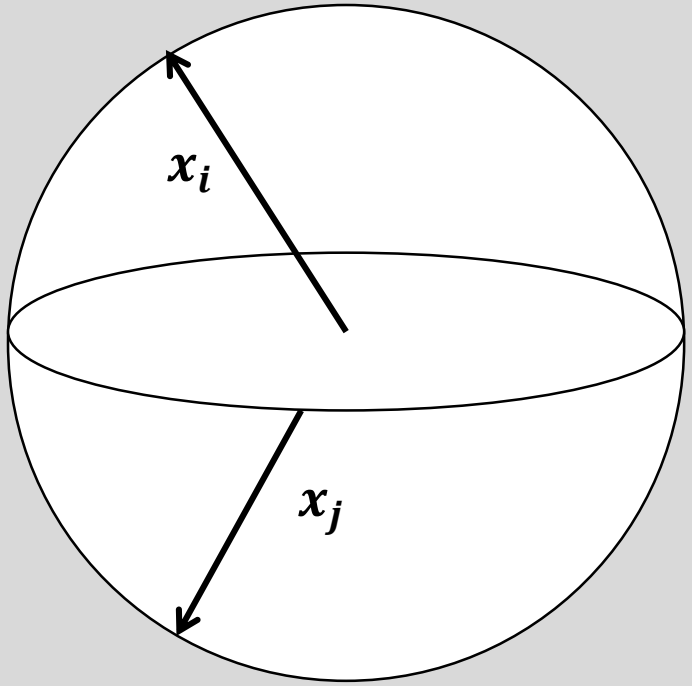


Our private algorithm uses sample problem instances to find a **nearly optimal greedy heuristic** for the specific application domain.

No **sensitive information** about the training set is revealed.

We show that our private algorithm has strong utility guarantees for many **algorithm configuration problems**.

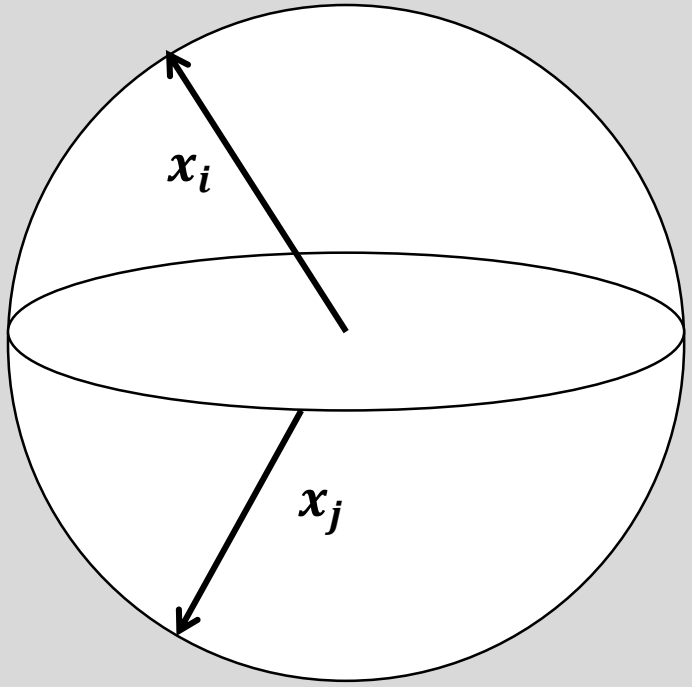
Integer quadratic programming algorithm configuration



IQPs are used in many applications where privacy preservation is essential, such as financial portfolio optimization.

We show that our private algorithm has strong utility guarantees for many **algorithm configuration problems**.

Integer quadratic programming algorithm configuration

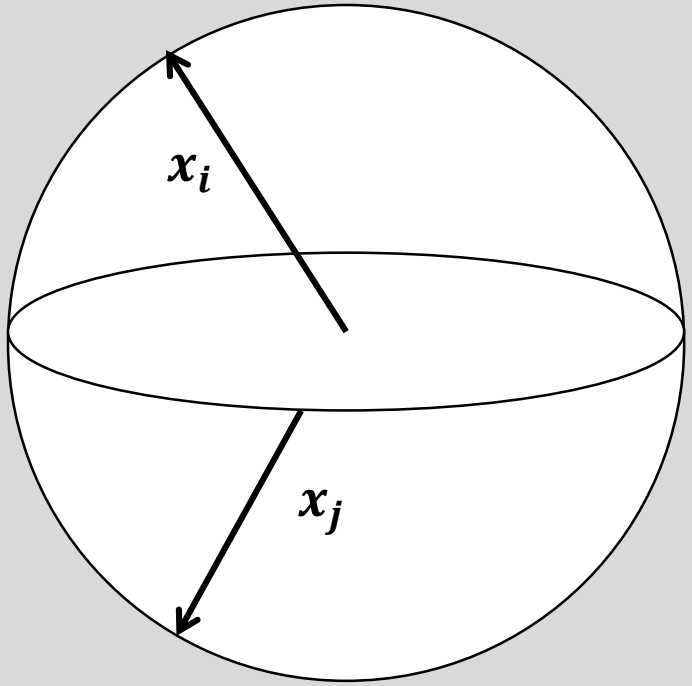


IQPs are often approximated by solving a semi-definite program and **rounding** the vectors to integer values.

There are many different rounding schemes with varying quality.

We show that our private algorithm has strong utility guarantees for many **algorithm configuration problems**.

Integer quadratic programming algorithm configuration



Our private algorithm uses sample IQP instances to find a **nearly optimal rounding scheme** for the specific application domain.

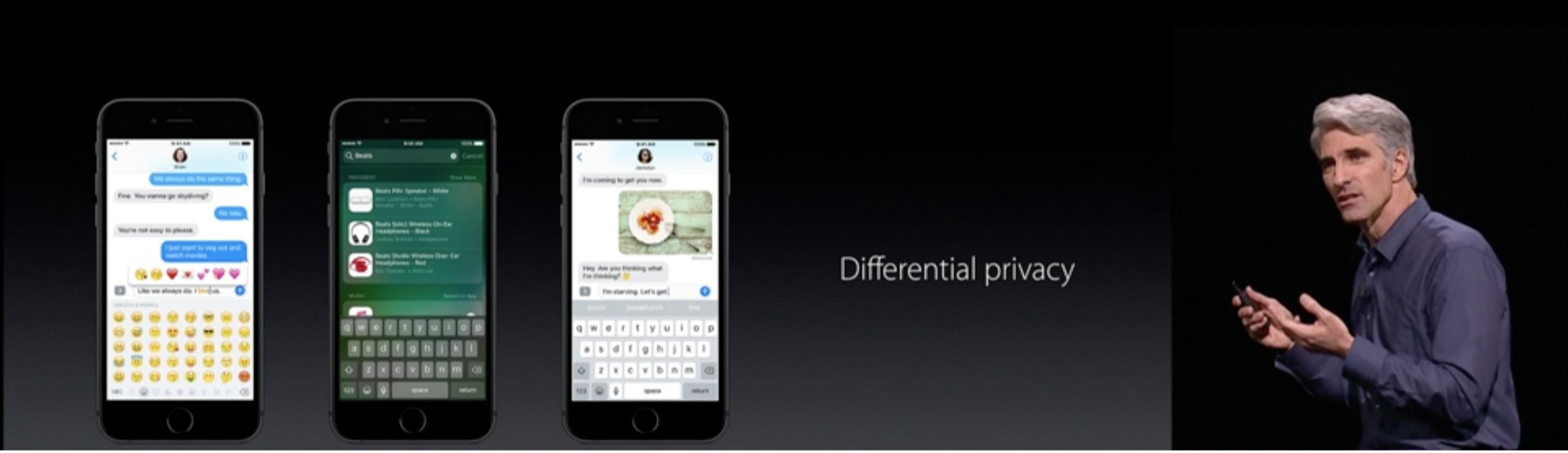
No **sensitive information** about the training set is revealed.

Learning-based mechanism design:
Use information about past consumers to design mechanisms that extract high revenue from future consumers.

Employed throughout industry.

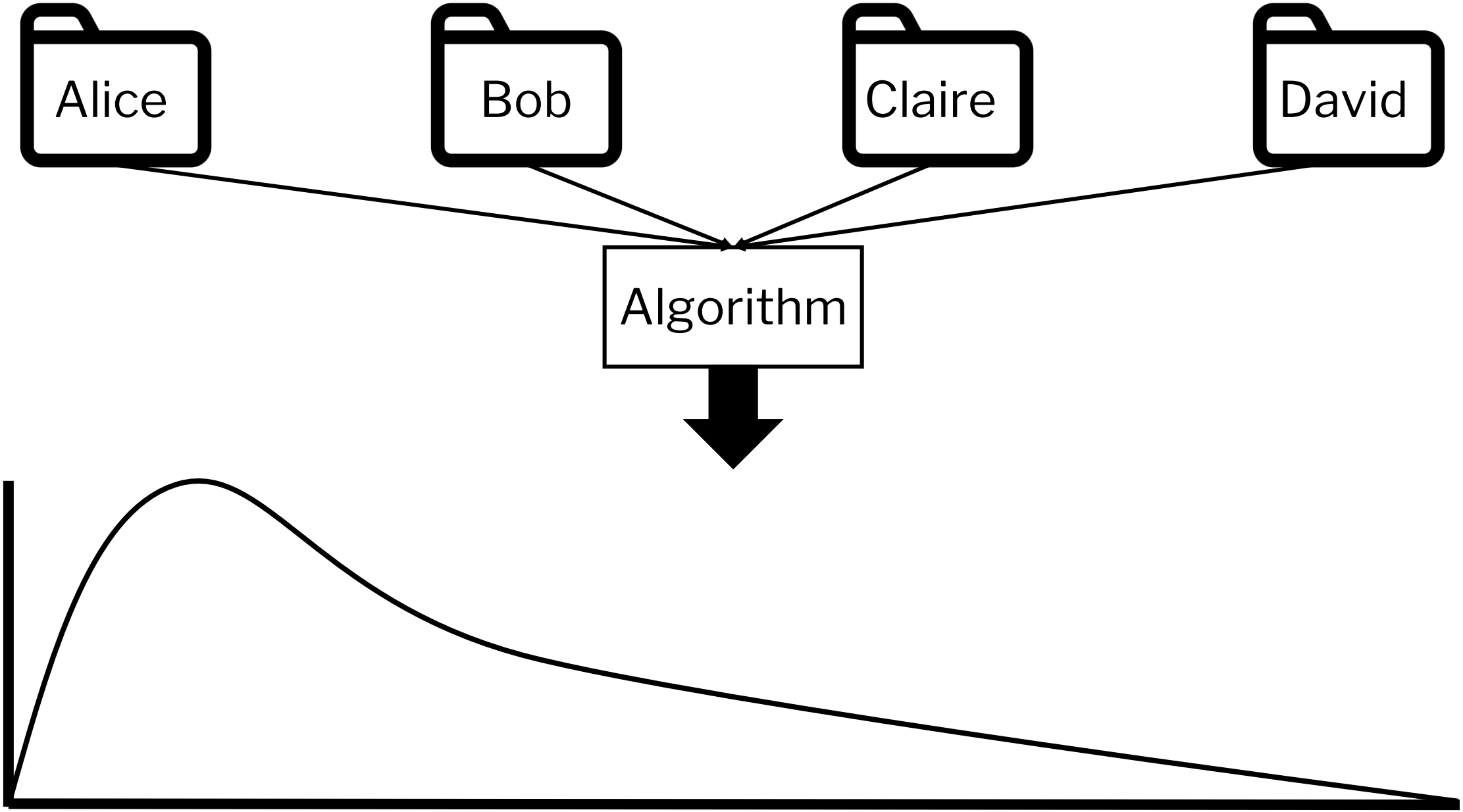
Garnered significant attention in TCS.
[Elkind, 2007, Cole and Roughgarden, 2014, Huang et al., 2015, Medina and Mohri, 2014, Morgenstern and Roughgarden, 2015, Devanur et al., 2016, etc.]

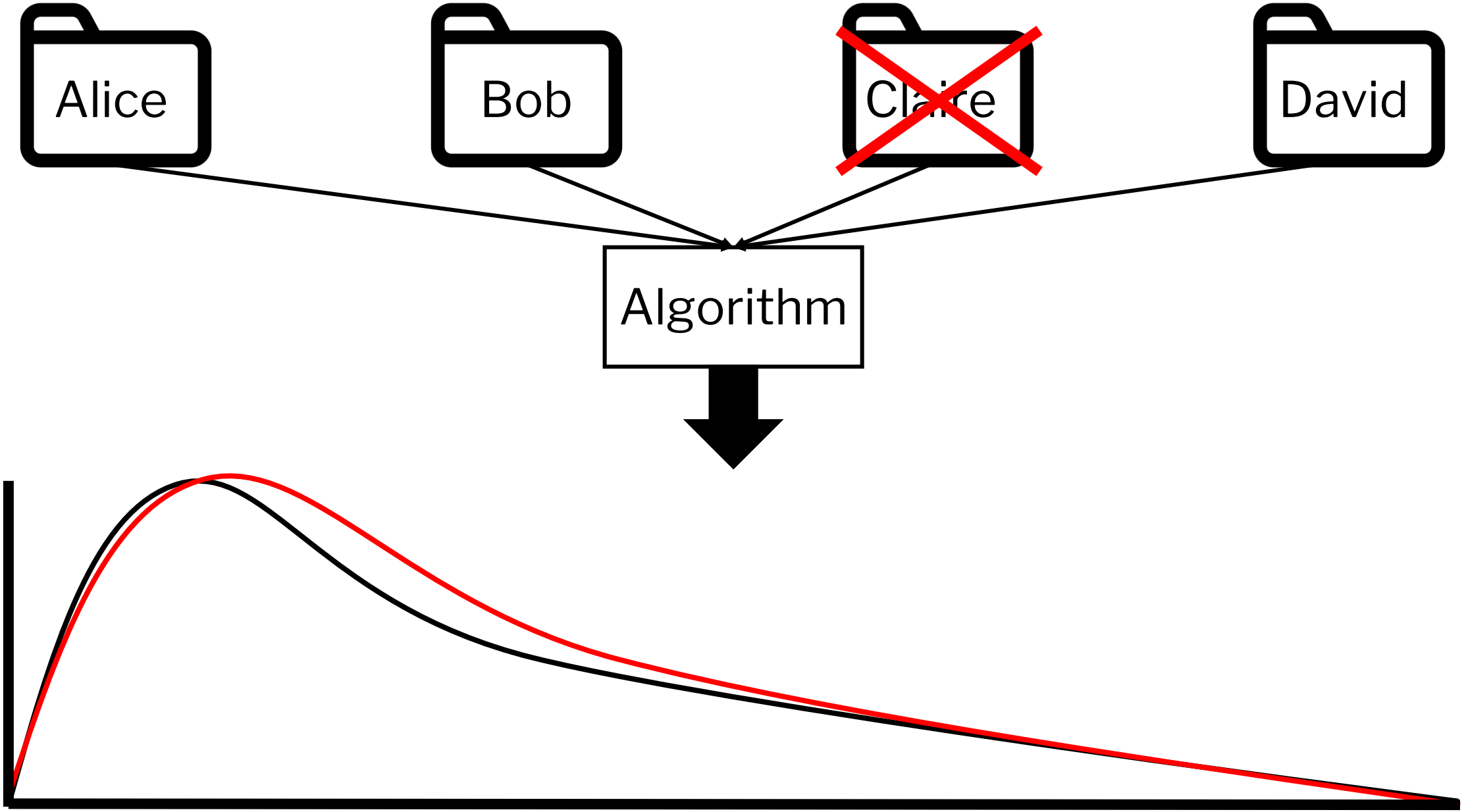
- Introduction
- Setup
  - Overview: Algorithm configuration and auction design
  - ➡ Differential privacy
  - Private pricing design
- The algorithm
  - Privacy guarantees
  - Utility guarantees
    - Example: private multi-item pricing
- Summary



An algorithm, given an input dataset D, is **differentially private** if the following holds:

The output reveals (almost) nothing more about a record in D than the output would have if the record wasn't contained in D.

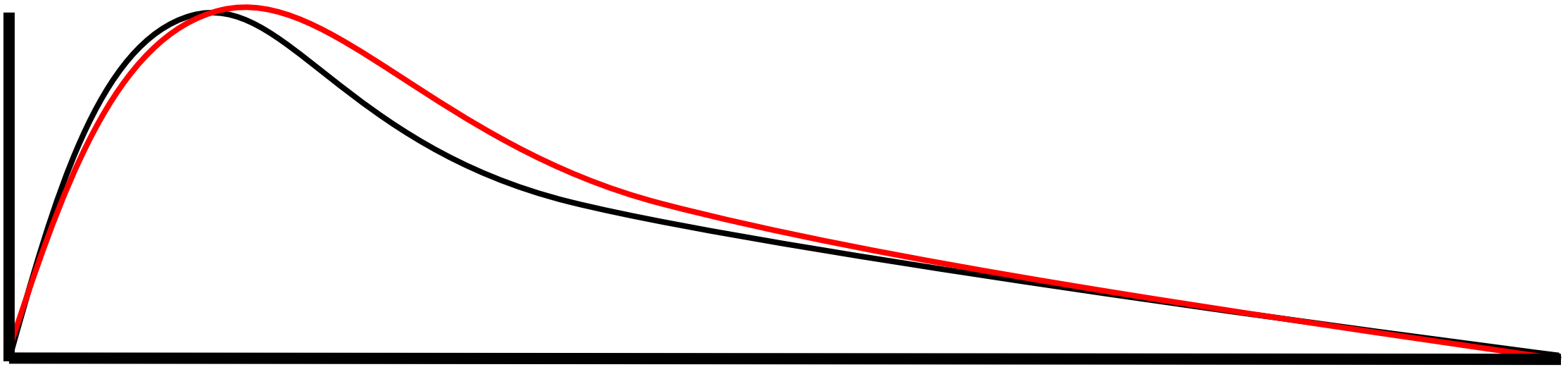

Alice
Bob
Claire
David
Algorithm


Alice
Bob
Claire
David
Algorithm

An algorithm $\mathcal{A}$ is $(\boldsymbol{\varepsilon}, \boldsymbol{\delta})$**-differentially private** if for all pairs of neighboring datasets D, D' and all sets $\mathcal{O}$ of outputs,

$$\mathbb{P}[\mathcal{A}(D) \in \mathcal{O}] \leq e^{\varepsilon}\mathbb{P}[\mathcal{A}(D') \in \mathcal{O}] + \delta$$

$e^{\varepsilon} \approx 1 + \varepsilon$



Introduction

Setup

- Overview: Algorithm configuration and auction design
- Differential privacy
- ➡ Private pricing design

The algorithm

- Privacy guarantees
- Utility guarantees
  - Example: private multi-item pricing

Summary

Single-item pricing problem:

One good for sale 

Single-item pricing problem:

Distribution over buyers:  $\sim \mathcal{D}$

Buyers' values are denoted as: value$\left(\right.$  ,  $\left.\right)$

Revenue$\left(\right.$  $, \rho\left.\right) = \rho \cdot \mathbf{1}\left(\text{value}\left(\right.\right.$  ,  $\left.\left.\right) \geq \rho\right)$

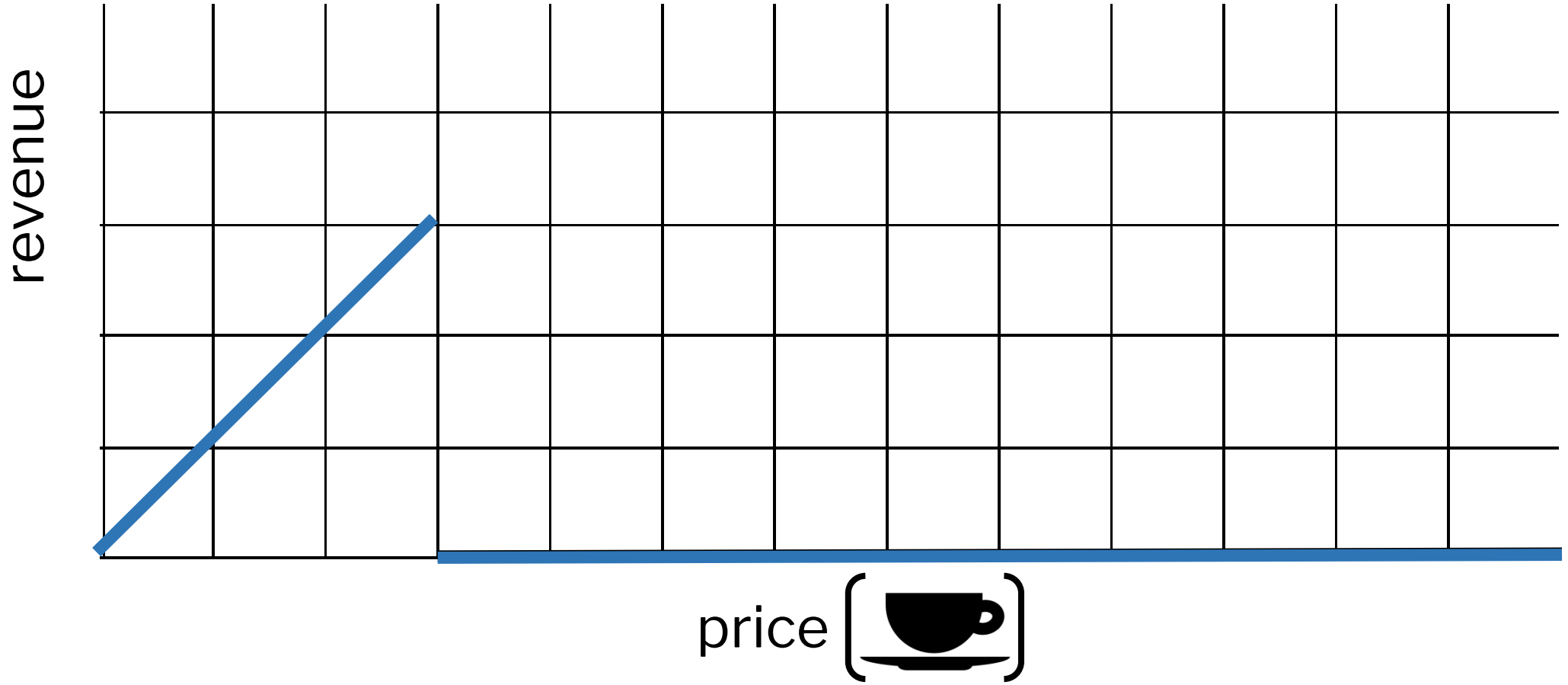
Pricing algorithm receives a set $\mathcal{S} = \{$  ,  , … ,  $\} \sim \mathcal{D}^N$

Goal: maximize $\frac{1}{N}\left(\text{Revenue}\left(\right.\right.$  $, \rho\left.\left.\right) + \cdots + \text{Revenue}\left(\right.\right.$  $, \rho\left.\left.\right)\right)$

Learning theory tells us that this value of ρ approximately maximizes $\mathbb{E}\left(\text{Revenue}\left(\right.\right.$  $, \rho\left.\left.\right)\right)$

$$S = \{ \text{person} \} \sim \mathcal{D}$$

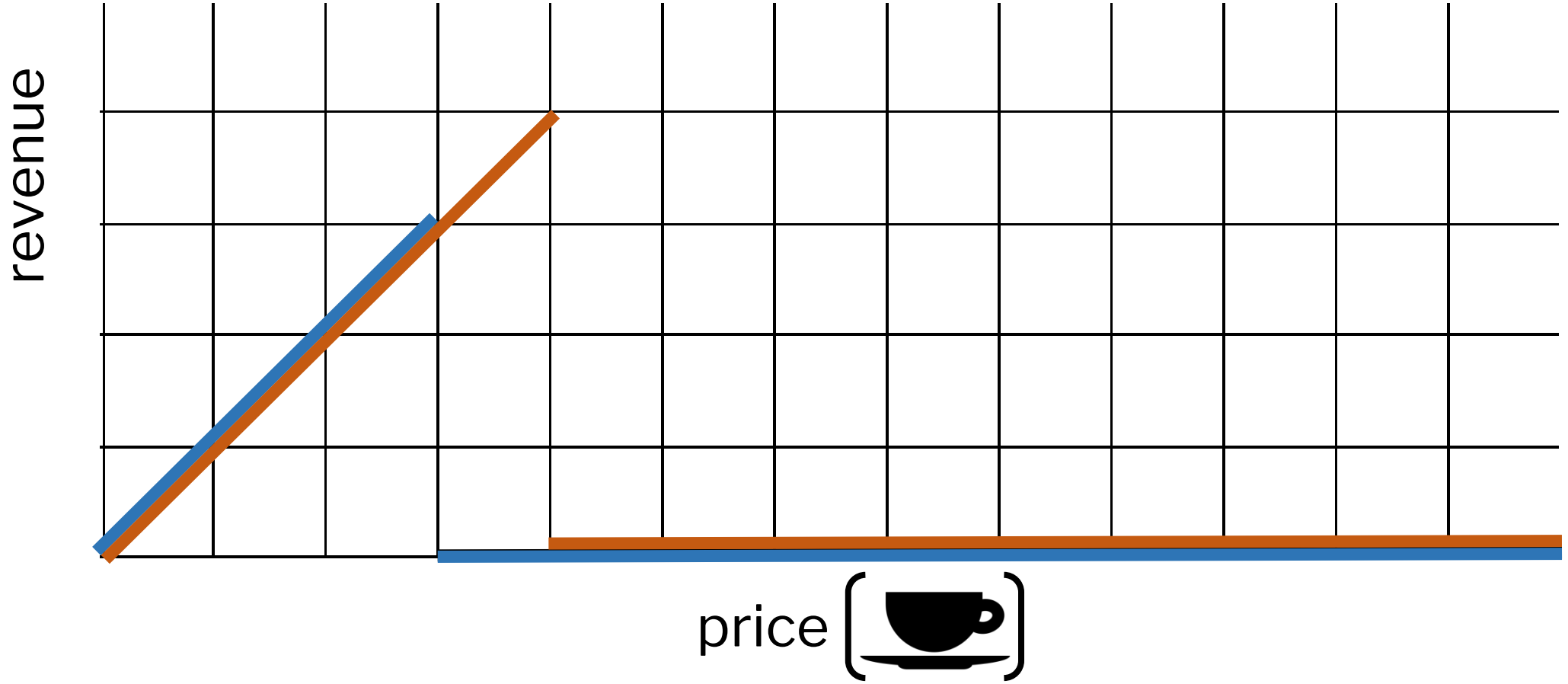
value(person, cup) = 3



$$\mathcal{S} = \{ \text{person}_1, \text{person}_2 \} \sim \mathcal{D}^2$$

value(person$_1$, cup) = 3

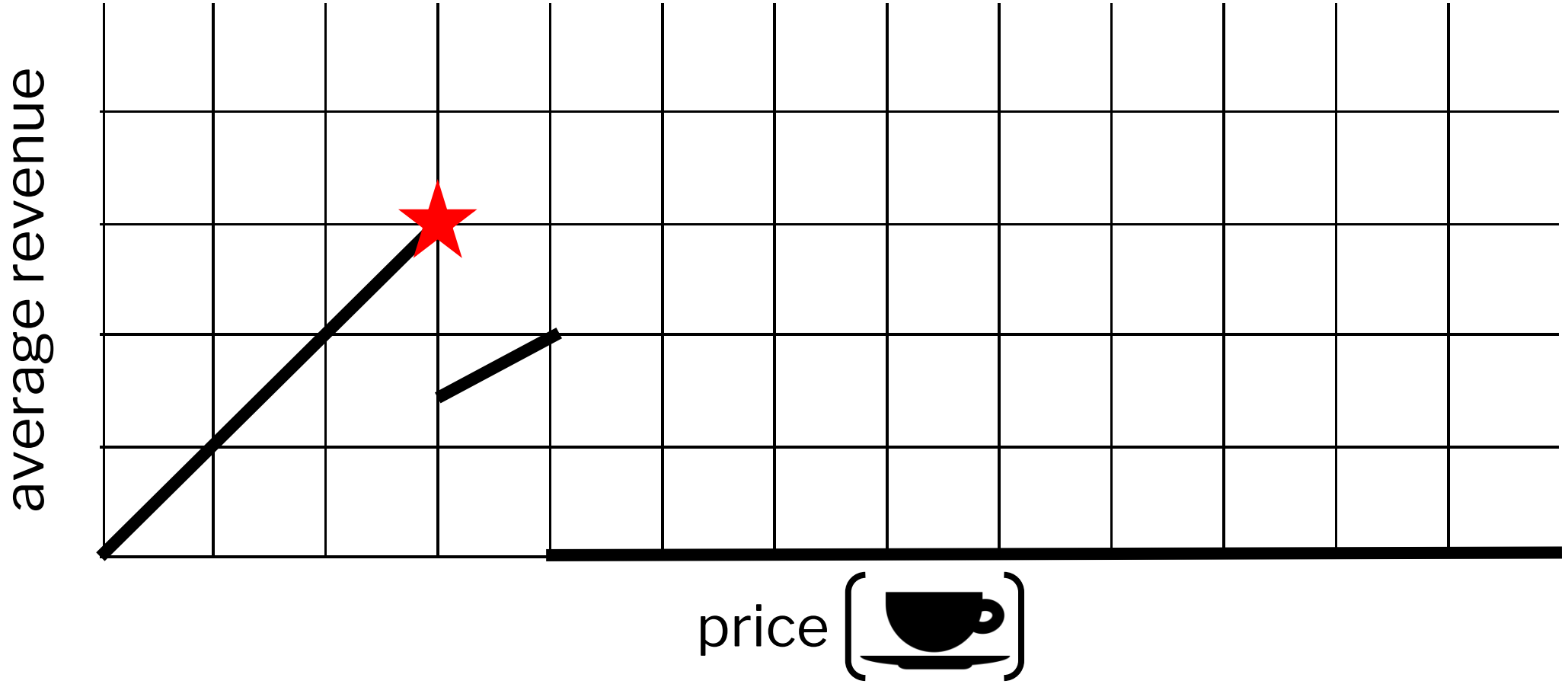
value(person$_2$, cup) = 4



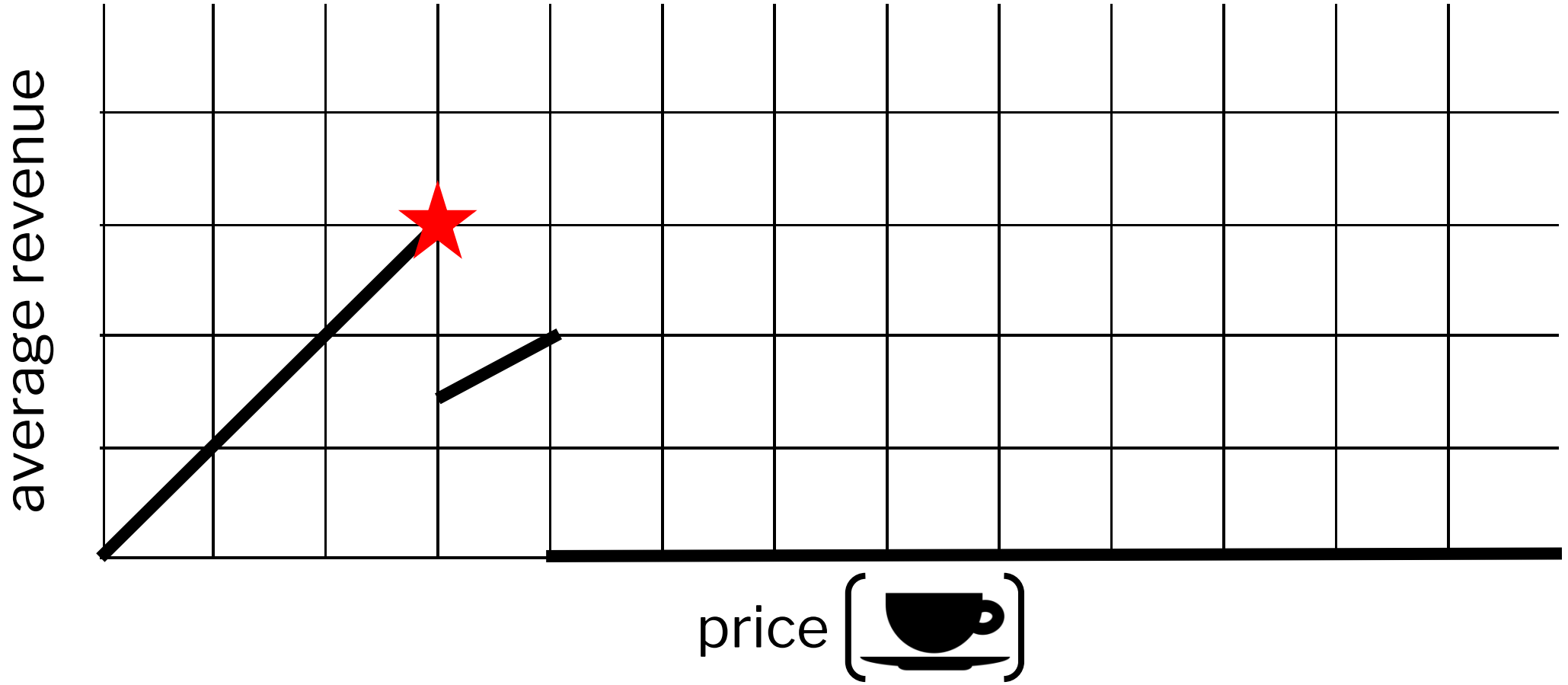
$$\mathcal{S} = \{ \text{person}, \text{person} \} \sim \mathcal{D}^2$$

value(person, cup) = 3

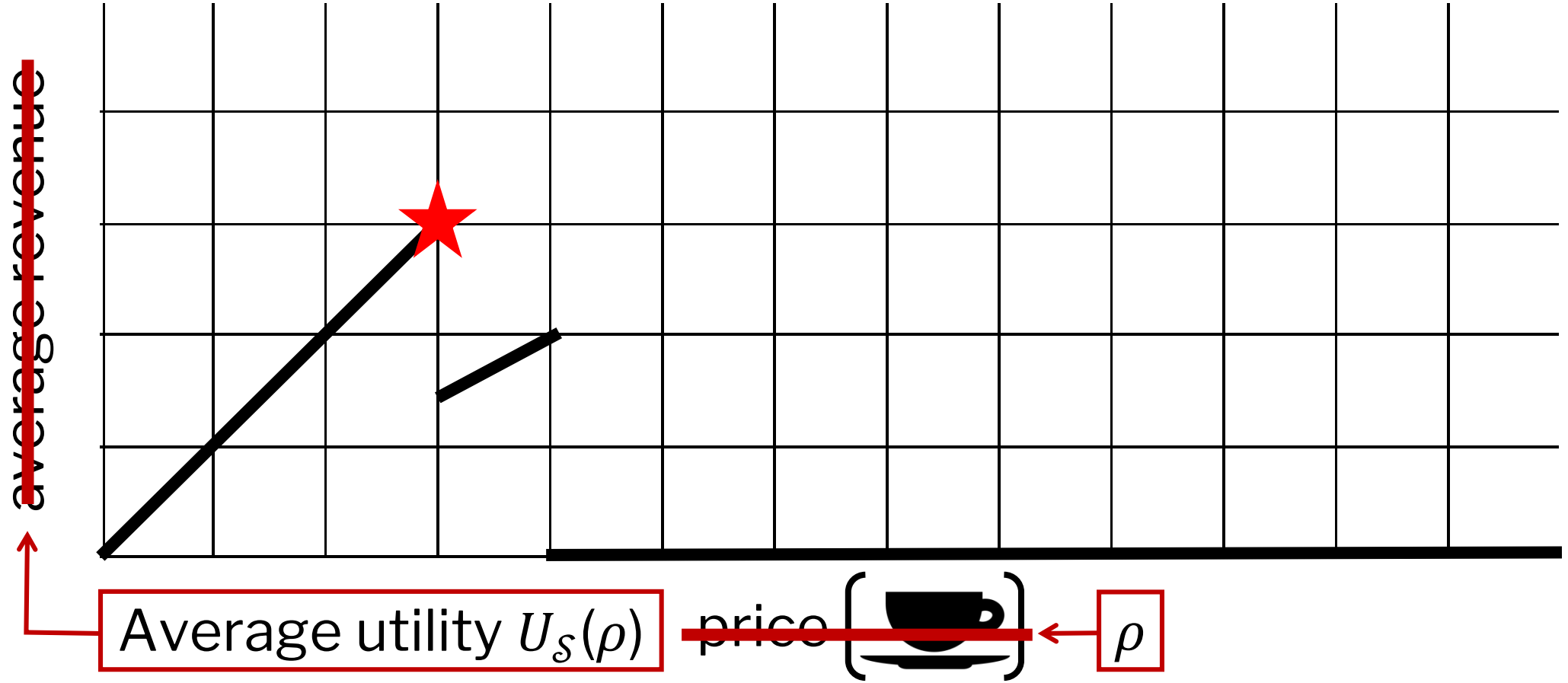
value(person, cup) = 4



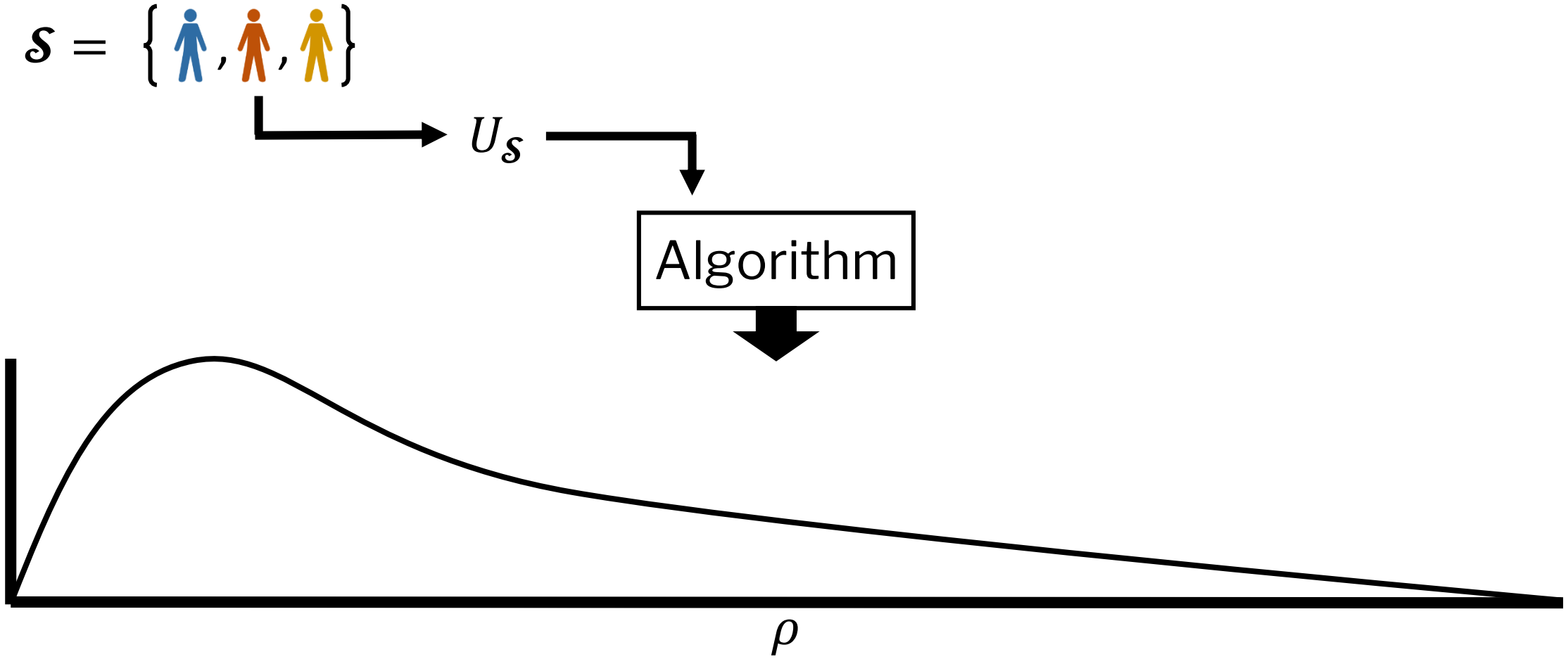
We want to write an algorithm that gets average revenue as close to $3 as possible while preserving differential privacy.



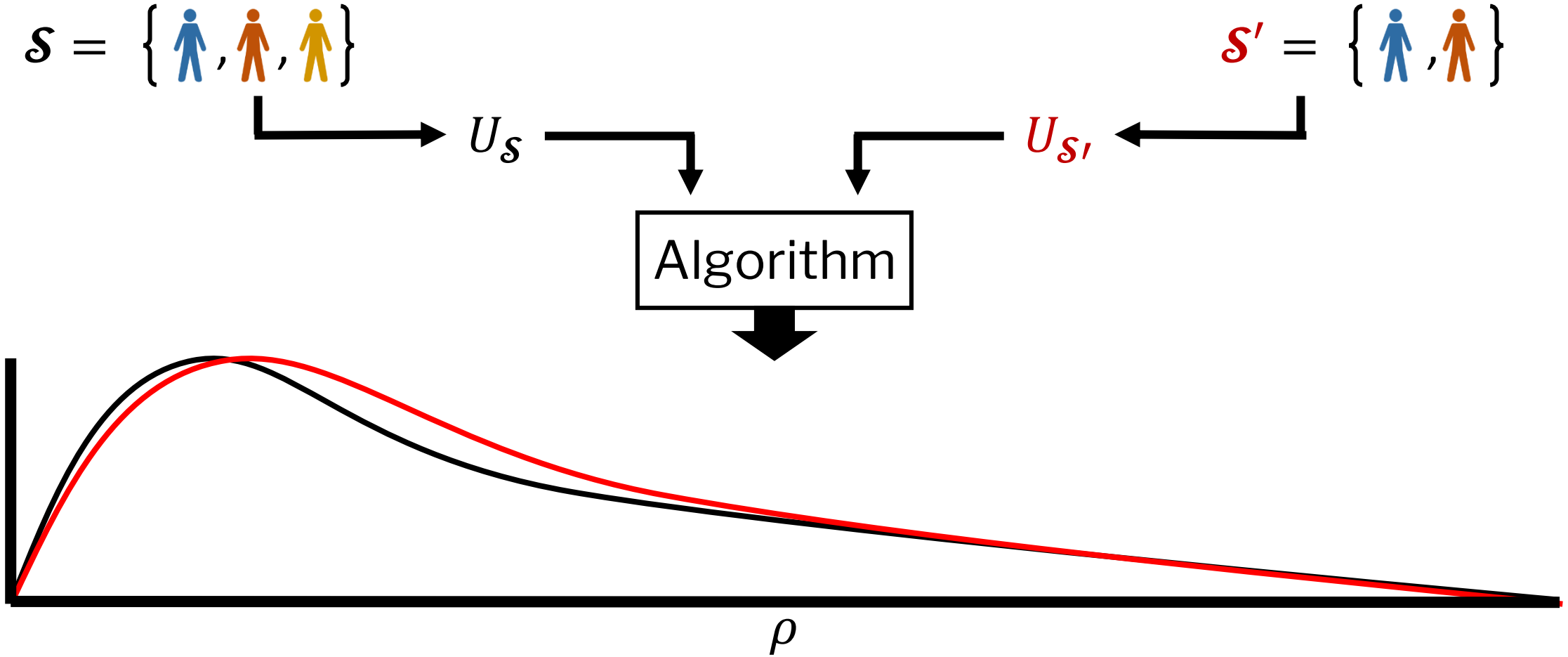
We want to write an algorithm that gets average revenue as close to $3 as possible while preserving differential privacy.



General problem:
Given a piecewise utility function $U_S(\rho)$, privately find a parameter $\hat{\rho}$ that approximately maximizes $U_S(\rho)$.



General problem:
Given a piecewise utility function $U_S(\rho)$, privately find a parameter $\hat{\rho}$ that approximately maximizes $U_S(\rho)$.



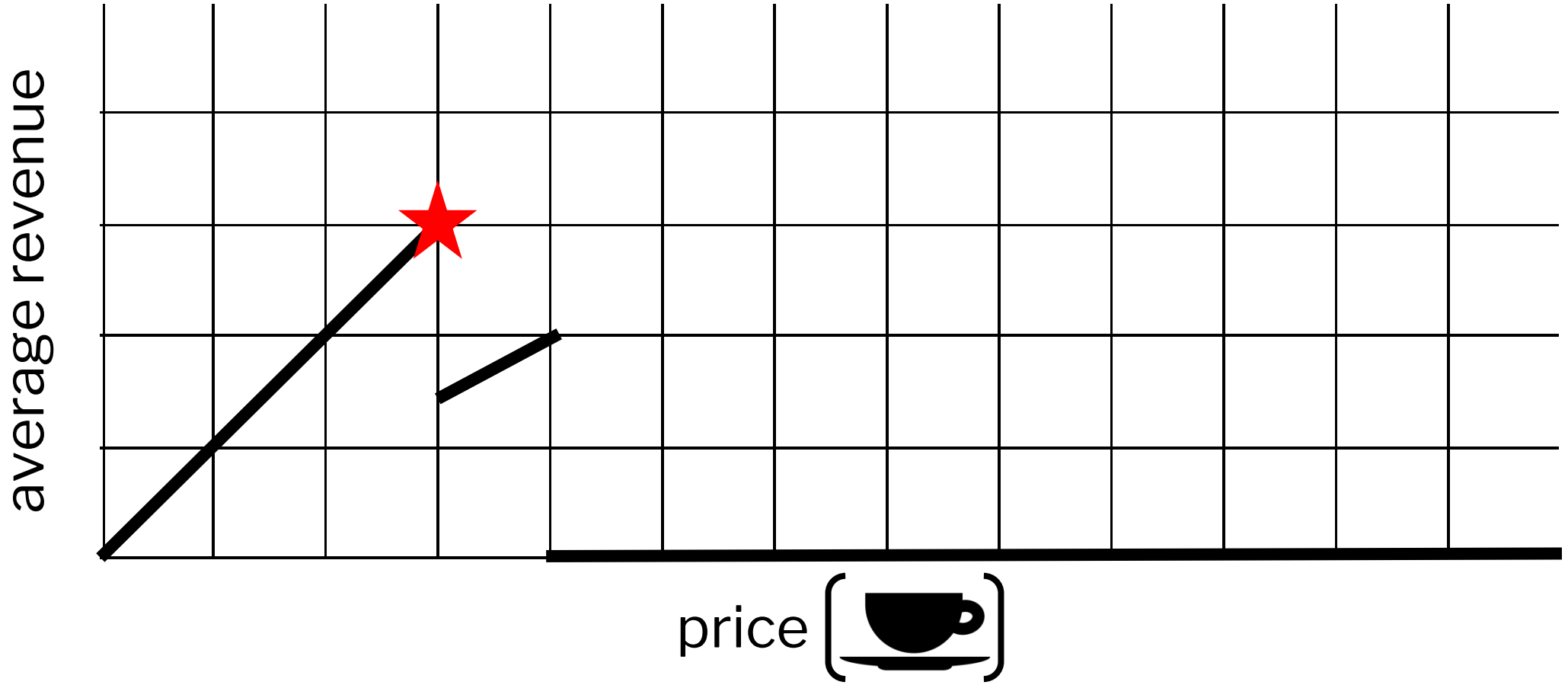
- Introduction
- Setup
  - Overview: Algorithm configuration and auction design
  - Differential privacy
  - Private pricing design
- ➡ The algorithm
  - Privacy guarantees
  - Utility guarantees
    - Example: private multi-item pricing
- Summary

Ideally, we want to use the exponential mechanism [McSherry and Talwar 2007]

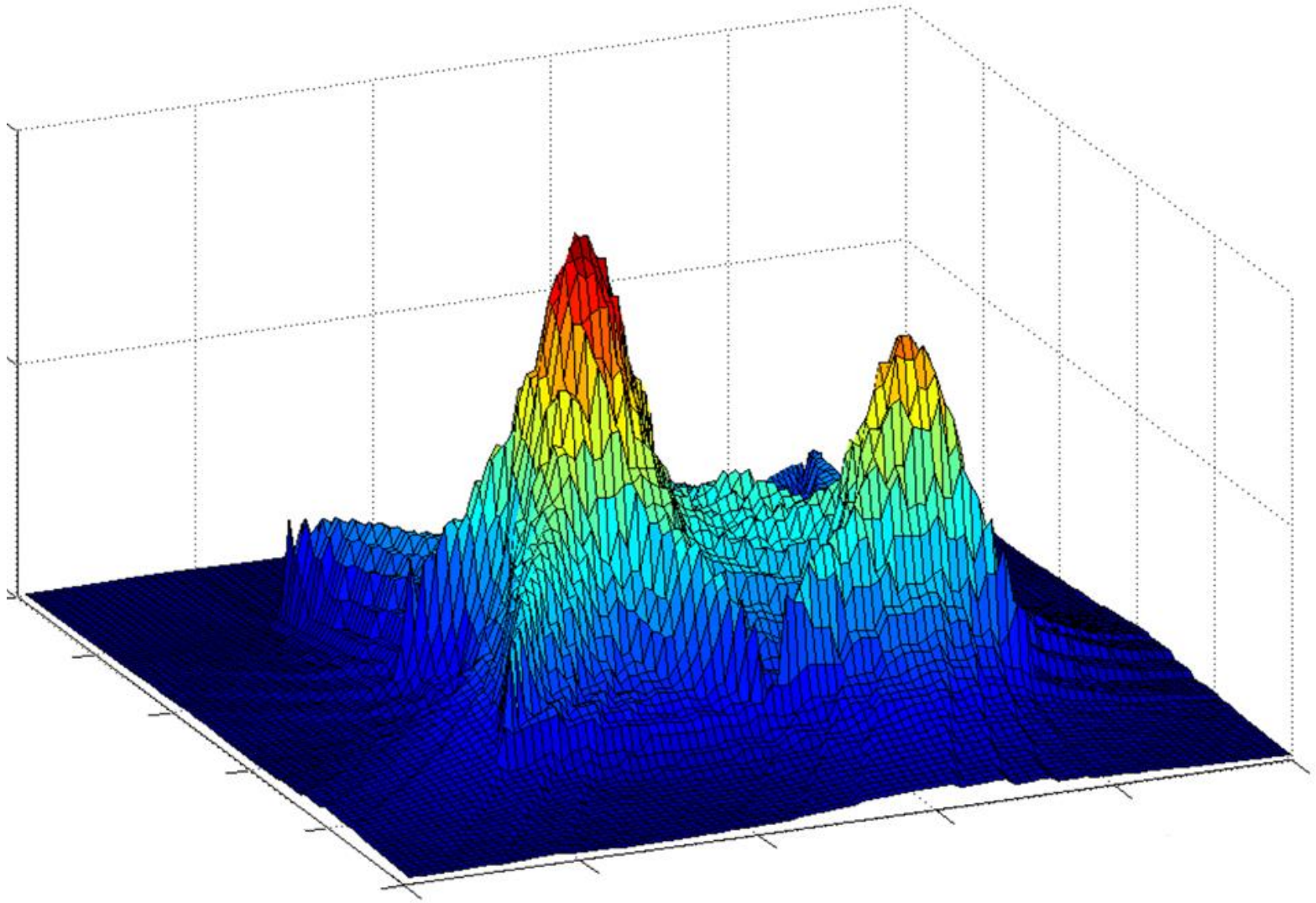
Pick a parameter vector $\boldsymbol{\rho}$ with probability proportional to

$$\exp\left(c \cdot \varepsilon \cdot U_S(\boldsymbol{\rho})\right)$$

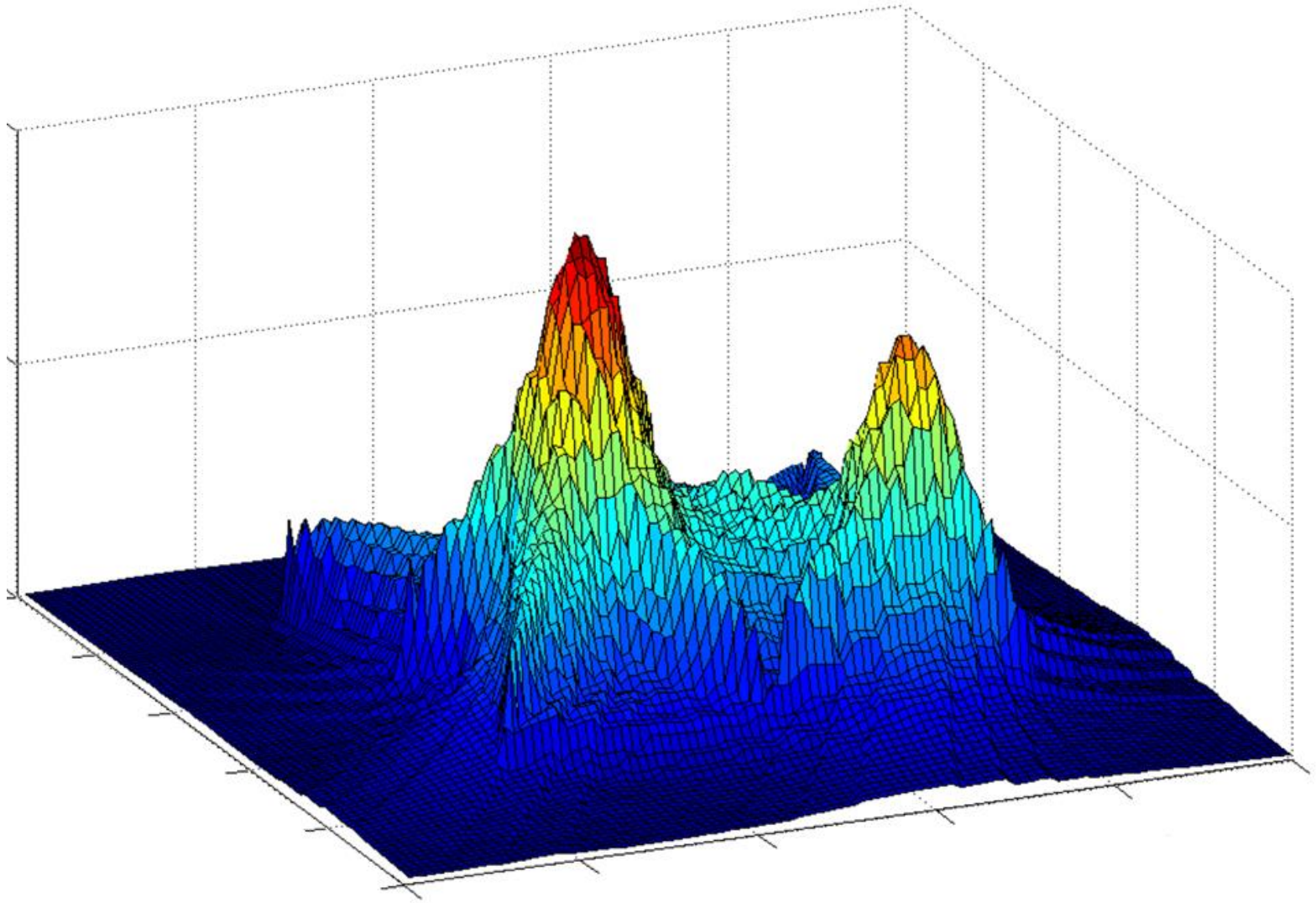
Sampling from $\exp\left(c \cdot \varepsilon \cdot U_S(\rho)\right)$ is easy when we can calculate its antiderivative and the parameter space is one-dimensional.



What about when the parameter space is multi-dimensional?

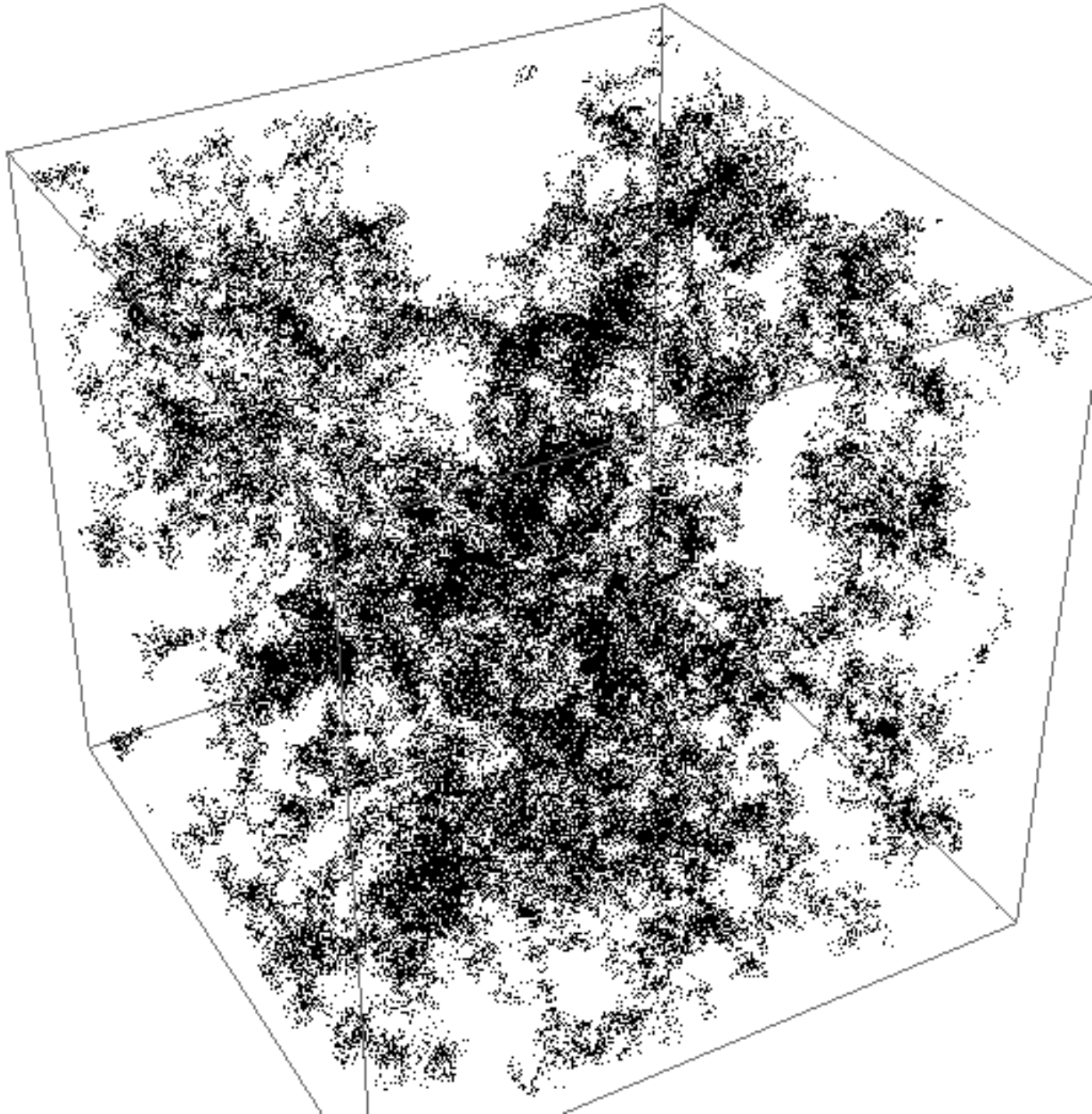


Generally, sampling from a multi-dimensional distribution is hard.



Existing efficient sampling techniques work for **logconcave distributions.** [Lovász and Vempala 2006, 2007, Bassily et al. 2014]

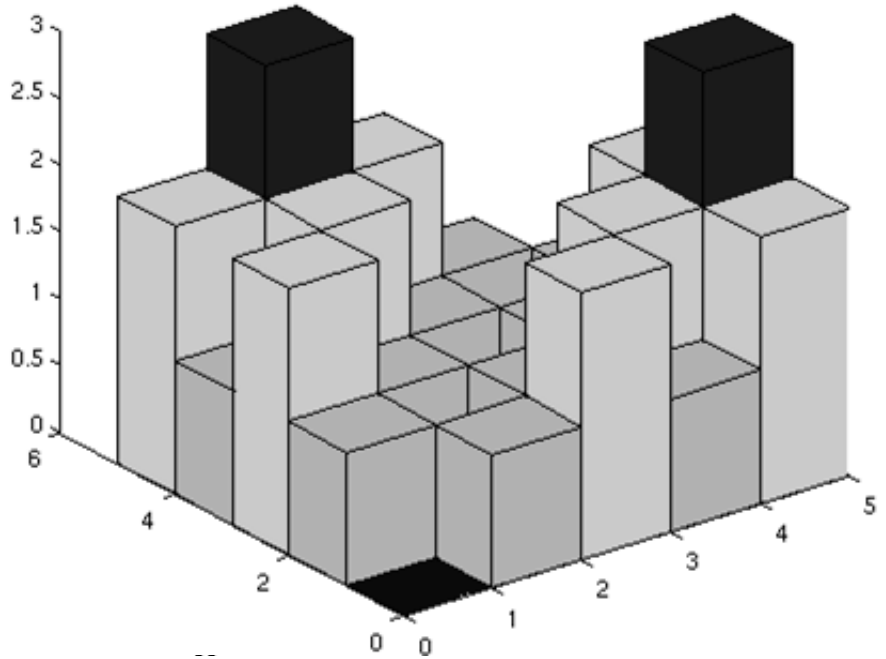
When $U_S(\boldsymbol{\rho})$ is piecewise concave, $\exp\big(c \cdot \varepsilon \cdot U_S(\boldsymbol{\rho})\big)$ is logconcave on each piece.



Theorem (part 1)

Suppose $U_{\mathcal{S}}(\boldsymbol{\rho})$ is piecewise L-Lipschitz and concave over convex regions in $\mathbb{R}^d$.

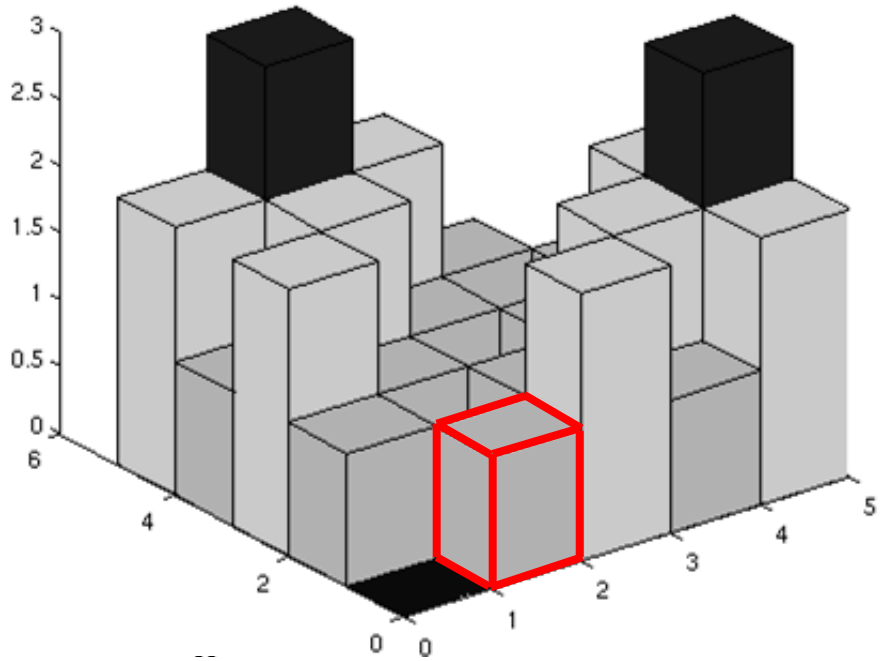
There is an (ε, δ)-DP algorithm that approximately maximizes $U_{\mathcal{S}}(\boldsymbol{\rho})$. Its running time polynomial in d, L, ε^{-1}, $\log \delta^{-1}$, and $|\mathcal{S}|$.



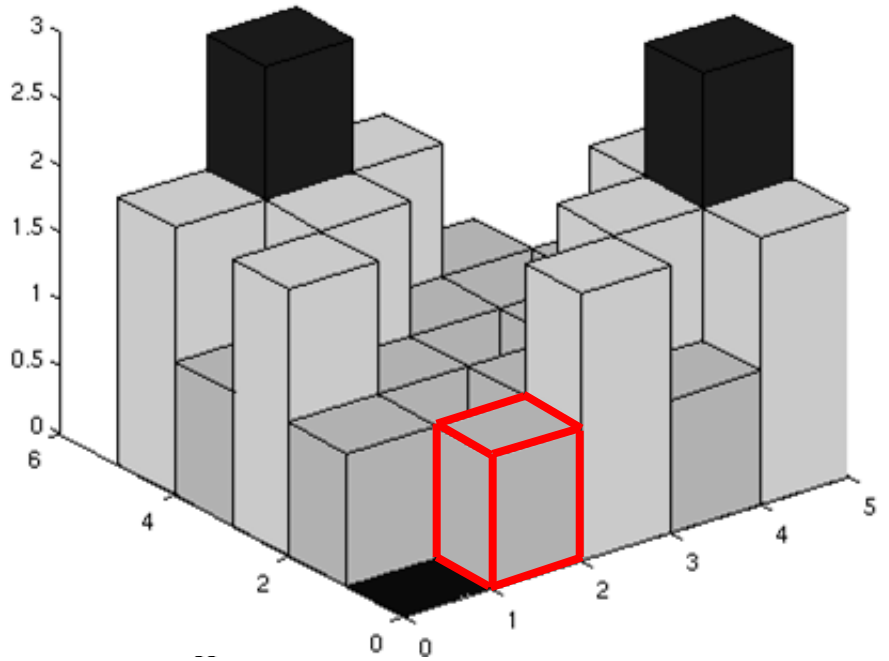
Step 1:

Estimate the probability mass that the exponential mechanism places on each piecewise portion of the domain.

This requires an integral estimation algorithm. [Lovász and Vempala ‘07]



Step 2:
Sample a portion of the domain according to that probability.

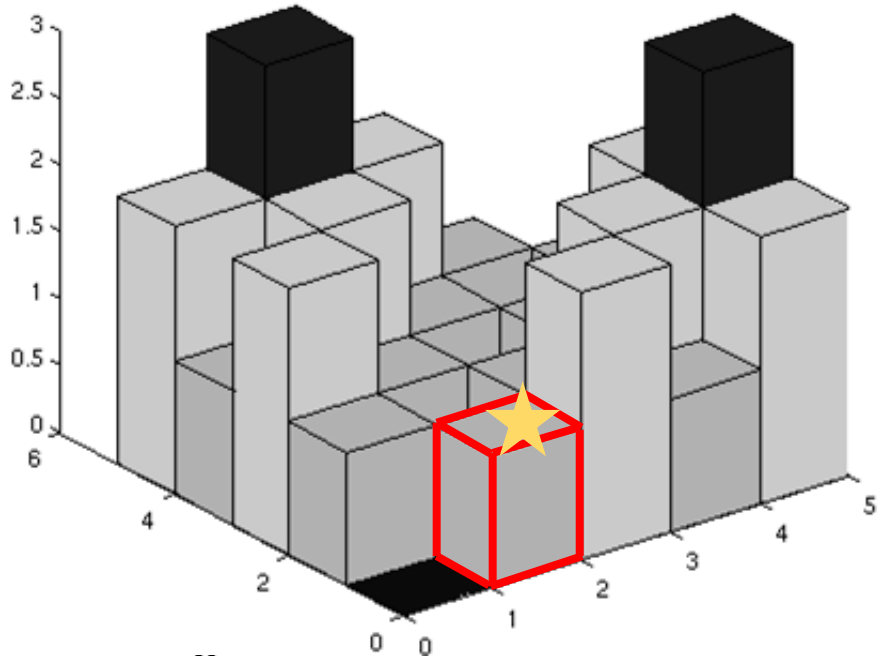


Step 3:

Sample a point in that portion of the domain with probability approximately according to the exponential mechanism:

$\exp\left(c \cdot \varepsilon \cdot U_S(\boldsymbol{\rho})\right)$

This requires approximate sampling [Bassily et al. 2014]



Proof sketch: Show that the output distribution is **almost** $\exp\left(c \cdot \varepsilon \cdot U_S(\boldsymbol{\rho})\right)$.

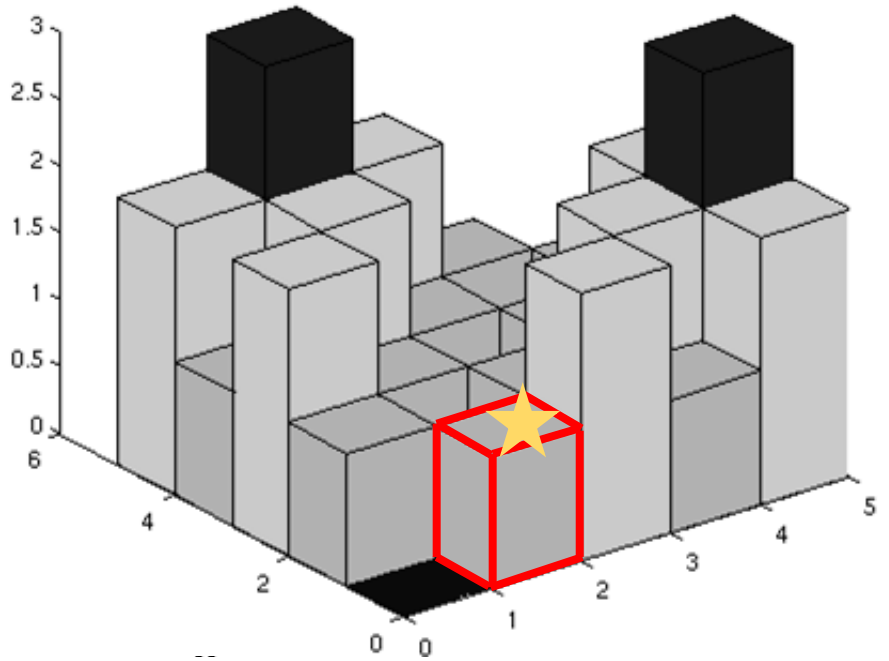
Lemma [Bassily et al. 2014]
If an algorithm $\mathcal{A}$'s output distribution is $\tilde{\varepsilon}$-close to that of a $(\varepsilon, 0)$-differentially private algorithm for every input dataset S, then $\mathcal{A}$ is $(2\tilde{\varepsilon} + \varepsilon, 0)$-differentially private. (Close under the metric $D_\infty\left(\mu_f, \mu_g\right) = \sup\lvert\log f(x)/g(x)\rvert$.)

Since the exponential mechanism is $(\varepsilon, 0)$-differentially private, our algorithm is $(O(\varepsilon), \boldsymbol{\delta})$-differentially private.

Theorem (part 1)

Suppose $U_{\mathcal{S}}(\boldsymbol{\rho})$ is piecewise L-Lipschitz and concave over convex regions in $\mathbb{R}^d$.

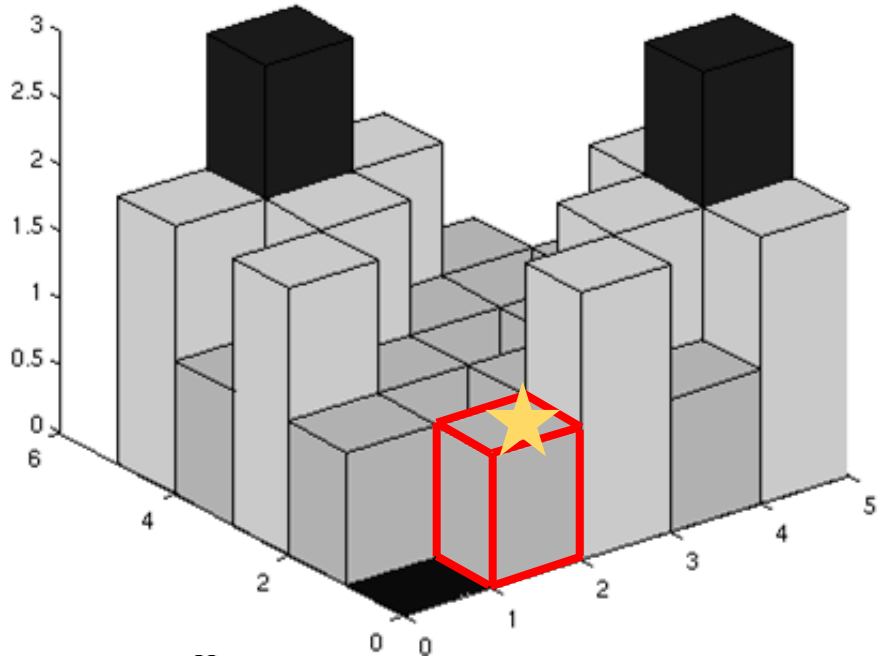
There is an (ε, δ)-DP algorithm that approximately maximizes $U_{\mathcal{S}}(\boldsymbol{\rho})$. Its running time polynomial in d, L, ε^{-1}, $\log \delta^{-1}$, and $|\mathcal{S}|$.



Theorem (part 1)

Suppose $U_S(\boldsymbol{\rho})$ is piecewise L-Lipschitz and concave over convex regions in $\mathbb{R}^d$.

There is an (ε, δ)-DP algorithm that **approximately** maximizes $U_S(\boldsymbol{\rho})$. Its running time polynomial in d, L, ε^{-1}, $\log \delta^{-1}$, and $|\mathcal{S}|$.



- Introduction
- Setup
  - Overview: Algorithm configuration and auction design
  - Differential privacy
  - Private pricing design
- The algorithm
  - Privacy guarantees
  - ➡ Utility guarantees
    - Example: private multi-item pricing
- Summary

Theorem (part 2)

Suppose $U_S(\boldsymbol{\rho})$ is $(\boldsymbol{w}, \boldsymbol{k})$**-dispersed**, piecewise L-Lipschitz, and the parameter space is contained in a ball of radius R.

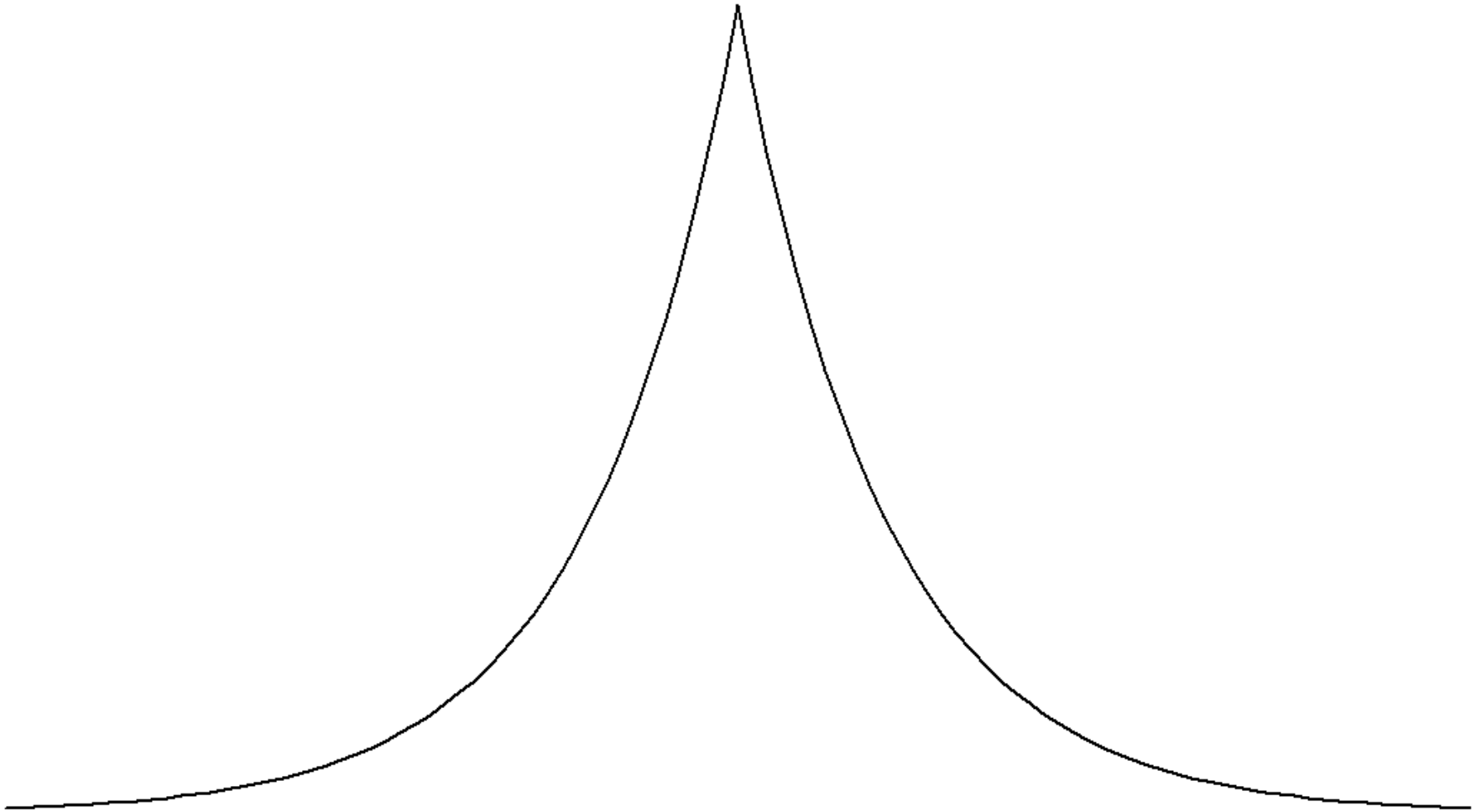
Let $\widehat{\boldsymbol{\rho}} \in \mathbb{R}^d$ be the parameter vector output by our (ε, δ)-DP algorithm.

With probability at least $1 - \gamma$,

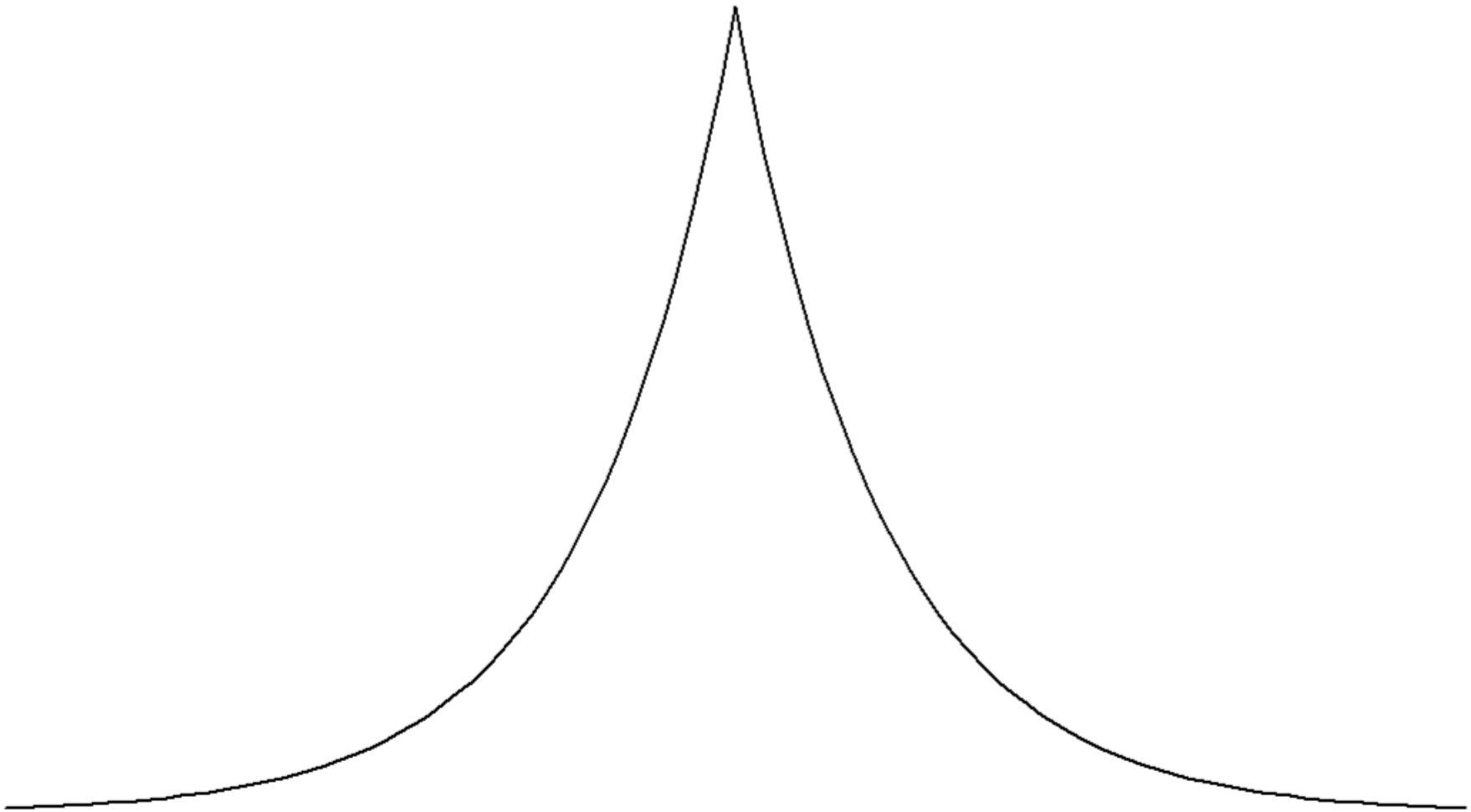
$$U_S(\widehat{\boldsymbol{\rho}}) \geq \max U_S(\boldsymbol{\rho}) - O\left(\frac{H}{|S|\varepsilon}\left(d \log\frac{R}{w} + \log\frac{1}{\gamma}\right) + Lw + \frac{Hk}{|S|}\right)$$

($U_S(\boldsymbol{\rho})$ is an average of functions with range in $[0, H]$.)

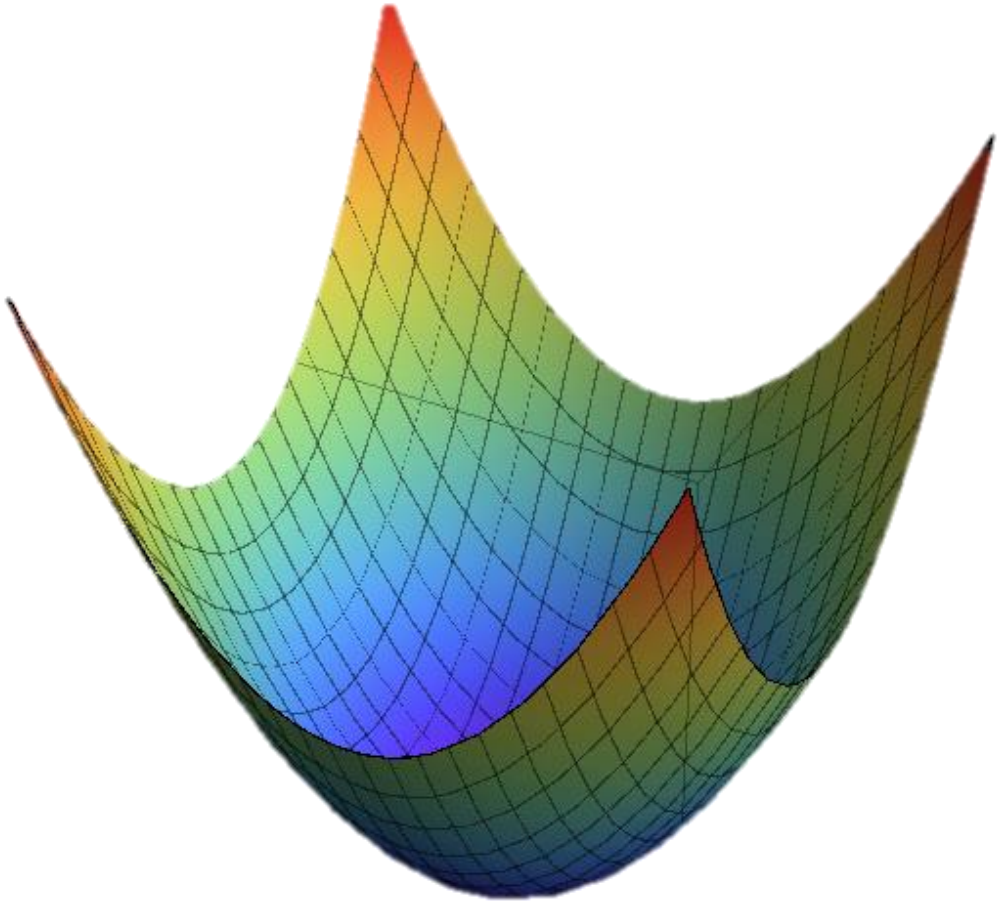
Every private algorithm needs to add some noise to the parameter it outputs.



Therefore, nearby parameters should have similar utilities.

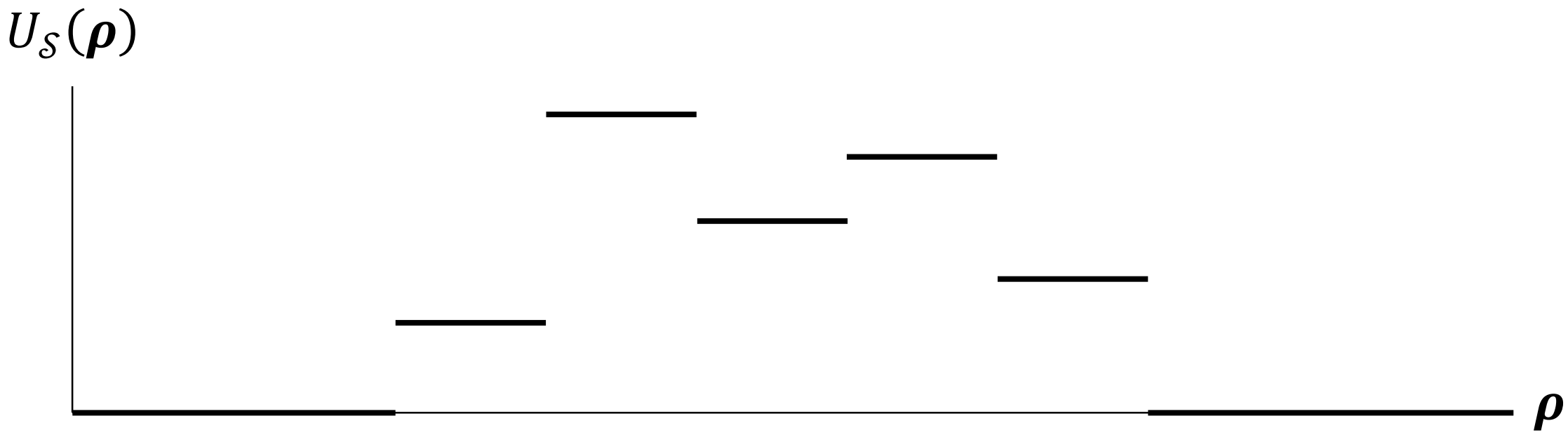


We often guarantee that nearby parameters have similar utilities by assuming the utility function is “nice.”



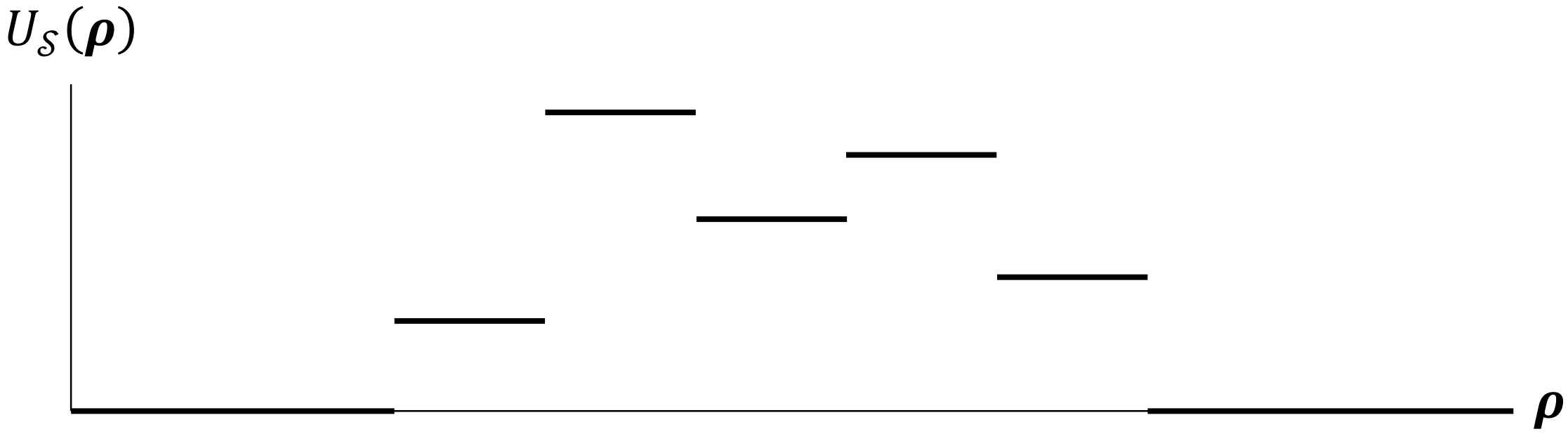
What does it mean for a piecewise function to be “nice”?

What does it mean for a piecewise function to be “nice”?



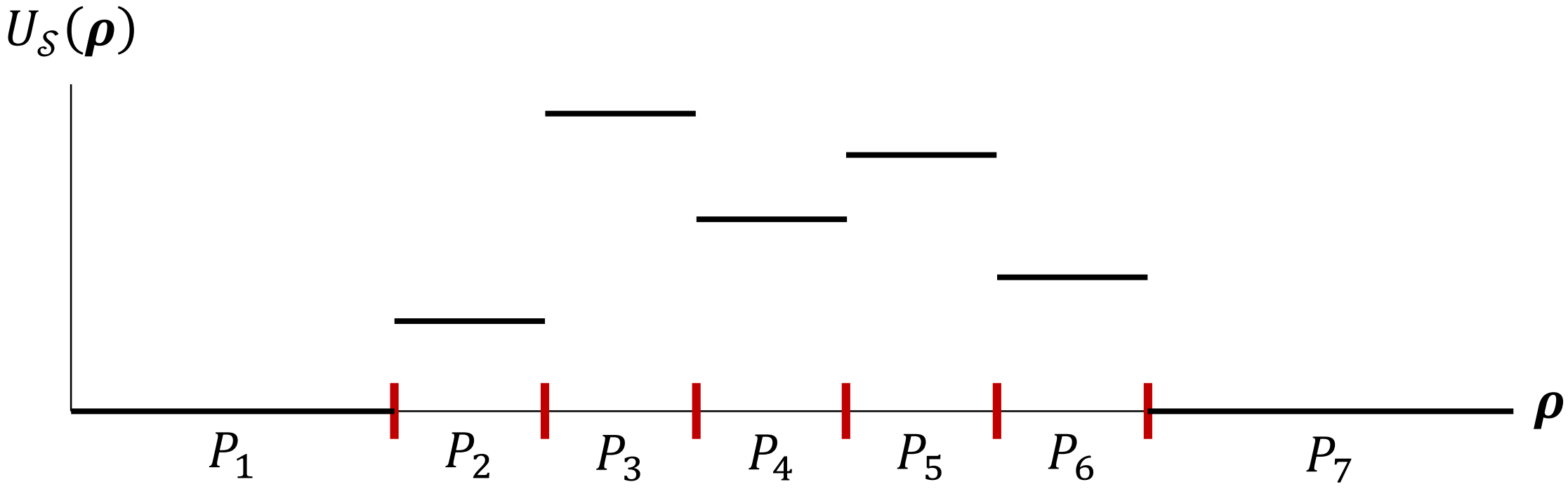
Let $\mathcal{P} = \{P_1, \ldots, P_t\}$ be a partition of the parameter space such that $U_S(\boldsymbol{\rho})$ is piecewise L-Lipschitz within each P_i.

U_S is **(w, k)-dispersed** if for any ball $\mathcal{B}$ of radius w, the number of sets in $\mathcal{P}$ that intersect $\mathcal{B}$ is at most k.



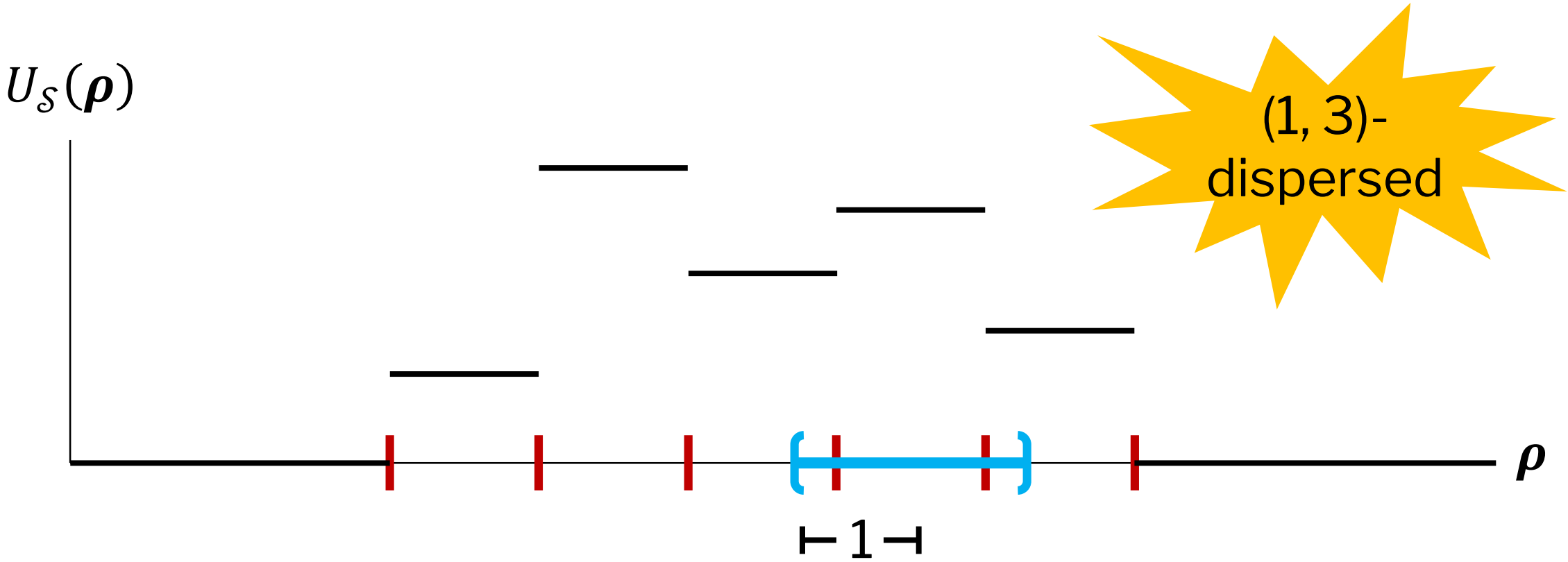
Let $\mathcal{P} = \{P_1, \ldots, P_t\}$ be a partition of the parameter space such that $U_S(\boldsymbol{\rho})$ is piecewise L-Lipschitz within each P_i.

U_S is **(w, k)-dispersed** if for any ball $\mathcal{B}$ of radius w, the number of sets in $\mathcal{P}$ that intersect $\mathcal{B}$ is at most k.



Let $\mathcal{P} = \{P_1, \ldots, P_t\}$ be a partition of the parameter space such that $U_S(\boldsymbol{\rho})$ is piecewise L-Lipschitz within each P_i.

U_S is **(w, k)-dispersed** if for any ball $\mathcal{B}$ of radius w, the number of sets in $\mathcal{P}$ that intersect $\mathcal{B}$ is at most k.



Theorem (part 2)

Suppose $U_S(\boldsymbol{\rho})$ is $(\boldsymbol{w}, \boldsymbol{k})$-dispersed, piecewise L-Lipschitz, and the parameter space is contained in a ball of radius R.

Let $\widehat{\boldsymbol{\rho}} \in \mathbb{R}^d$ be the parameter vector output by our (ε, δ)-DP algorithm.

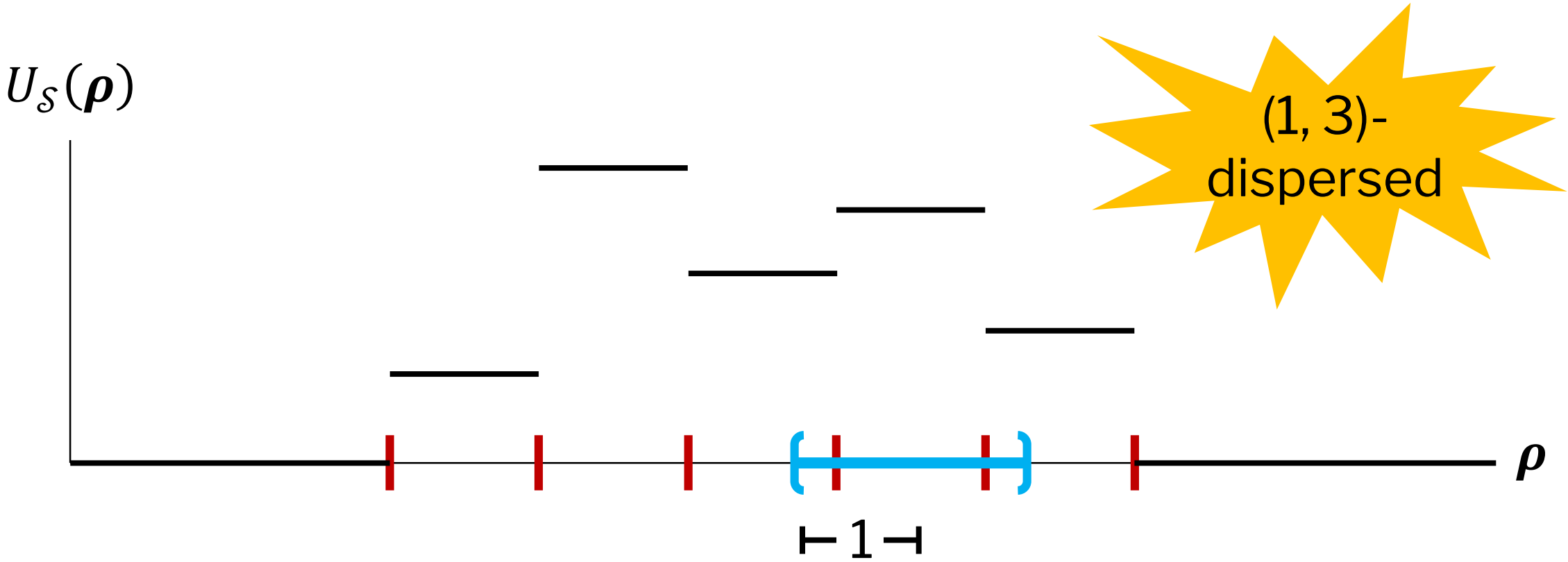
With probability at least $1 - \gamma$,

$$U_S(\widehat{\boldsymbol{\rho}}) \geq \max U_S(\boldsymbol{\rho}) - O\left(\frac{H}{|S|\varepsilon}\left(d \log \frac{R}{\boldsymbol{w}} + \log \frac{1}{\gamma}\right) + L\boldsymbol{w} + \frac{H\boldsymbol{k}}{|S|}\right)$$

($U_S(\boldsymbol{\rho})$ is an average of functions with range in $[0, H]$.)

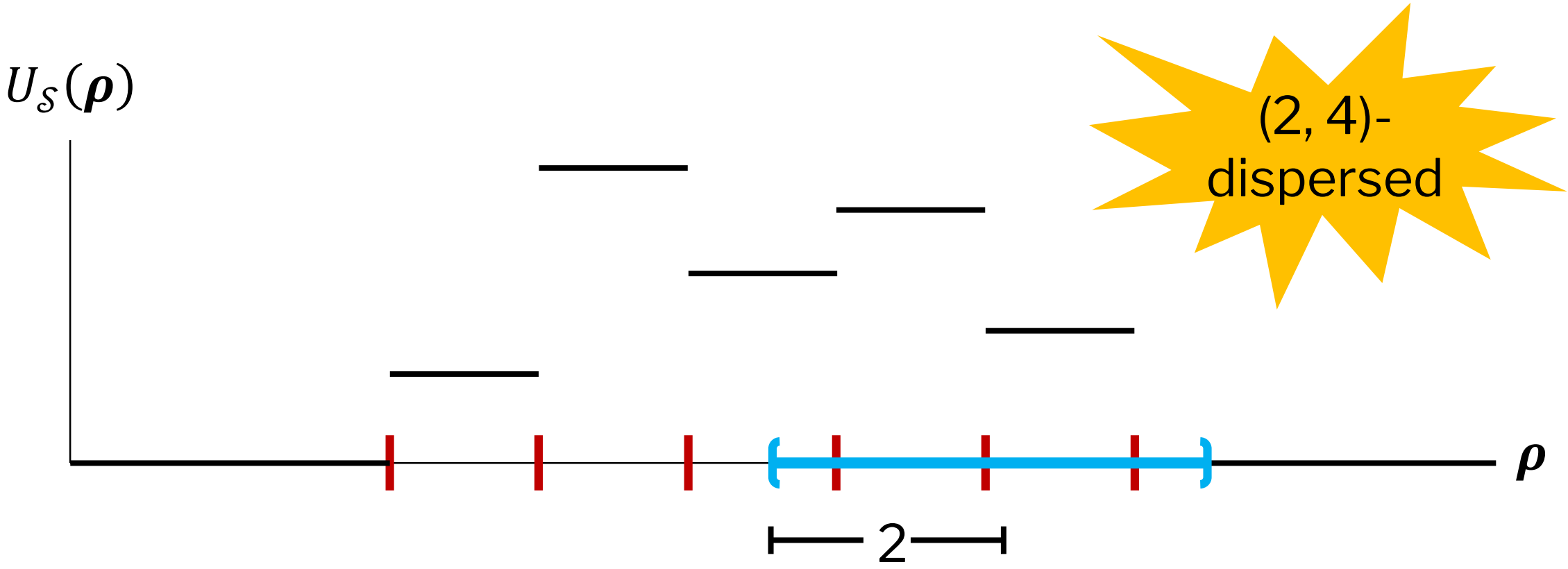
Let $\mathcal{P} = \{P_1, \ldots, P_t\}$ be a partition of the parameter space such that $U_S(\boldsymbol{\rho})$ is piecewise L-Lipschitz within each P_i.

U_S is **(w, k)-dispersed** if for any ball $\mathcal{B}$ of radius w, the number of sets in $\mathcal{P}$ that intersect $\mathcal{B}$ is at most k.



Let $\mathcal{P} = \{P_1, \ldots, P_t\}$ be a partition of the parameter space such that $U_S(\boldsymbol{\rho})$ is piecewise L-Lipschitz within each P_i.

U_S is $(\boldsymbol{w}, \boldsymbol{k})$**-dispersed** if for any ball $\mathcal{B}$ of radius w, the number of sets in $\mathcal{P}$ that intersect $\mathcal{B}$ is at most k.



Theorem (part 2)

Suppose $U_S(\boldsymbol{\rho})$ is $(\boldsymbol{w}, \boldsymbol{k})$-dispersed, piecewise L-Lipschitz, and the parameter space is contained in a ball of radius R.

Let $\widehat{\boldsymbol{\rho}} \in \mathbb{R}^d$ be the parameter vector output by our (ε, δ)-DP algorithm.

With probability at least $1 - \gamma$,

$$U_S(\widehat{\boldsymbol{\rho}}) \geq \max U_S(\boldsymbol{\rho}) - O\left(\frac{H}{|S|\varepsilon}\left(d \log \frac{R}{\boldsymbol{w}} + \log \frac{1}{\gamma}\right) + L\boldsymbol{w} + \frac{H\boldsymbol{k}}{|S|}\right)$$

($U_S(\boldsymbol{\rho})$ is an average of functions with range in $[0, H]$.)

- Introduction
- Setup
  - Overview: Algorithm configuration and auction design
  - Differential privacy
  - Private pricing design
- The algorithm
  - Privacy guarantees
  - Utility guarantees
  - ➡ Example: private multi-item pricing
- Summary

Multi-item pricing problem:

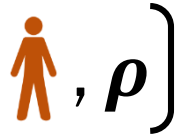
Multiple goods for sale 

Multi-item pricing problem:

Distribution over buyers: $\sim \mathcal{D}$

Buyers' values are denoted as: $\text{value}\left(\right.$  $\left.\right)$

$\text{value}\left(\right.$  $\left.\right)$

$$\text{Revenue}\left(\text{  }, \boldsymbol{\rho}\right) = \rho_1 \cdot \mathbf{1}\left[\text{value}\left(\text{  }\right) \geq \rho_1\right]$$

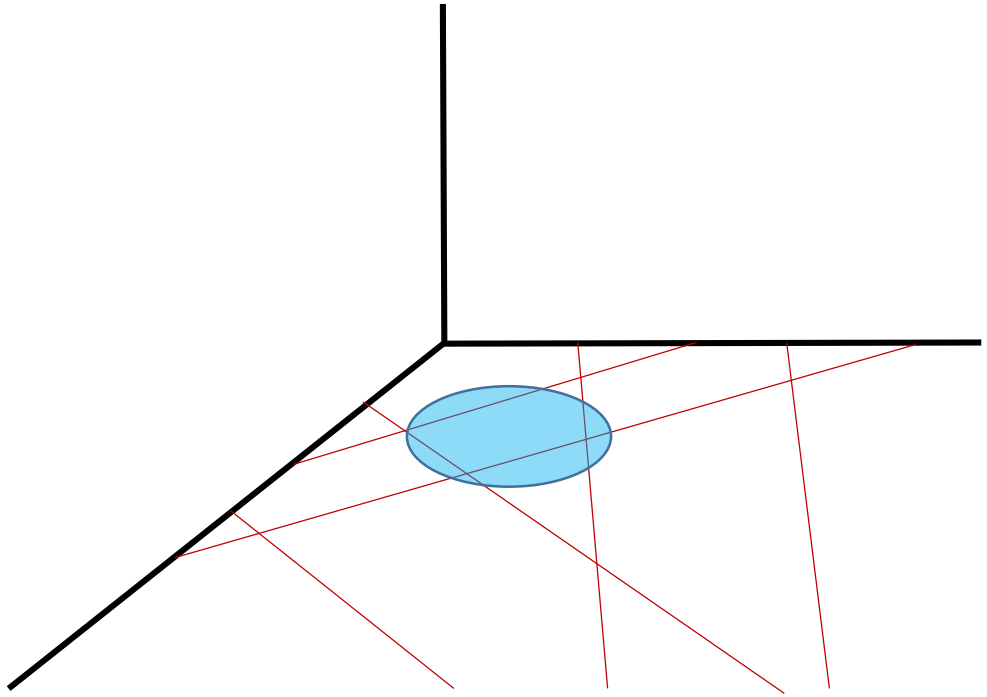
$$+ \rho_2 \cdot \mathbf{1}\left[\text{value}\left(\text{  }\right) \geq \rho_2\right]$$

Pricing algorithm receives a set $\mathcal{S} = \{$  , , … ,  $\} \sim \mathcal{D}^N$

Goal: maximize $\frac{1}{N}\left[\text{Revenue}\left(\text{  }, \boldsymbol{\rho}\right) + \cdots + \text{Revenue}\left(\text{  }, \boldsymbol{\rho}\right)\right]$

The average revenue over $\mathcal{S} = \{$  $\}$ of a pricing mechanism is piecewise linear and 1-Lipschitz.

$U_{\mathcal{S}}(\boldsymbol{\rho})$ = average revenue



ρ_1 = price 

ρ_2 = price 

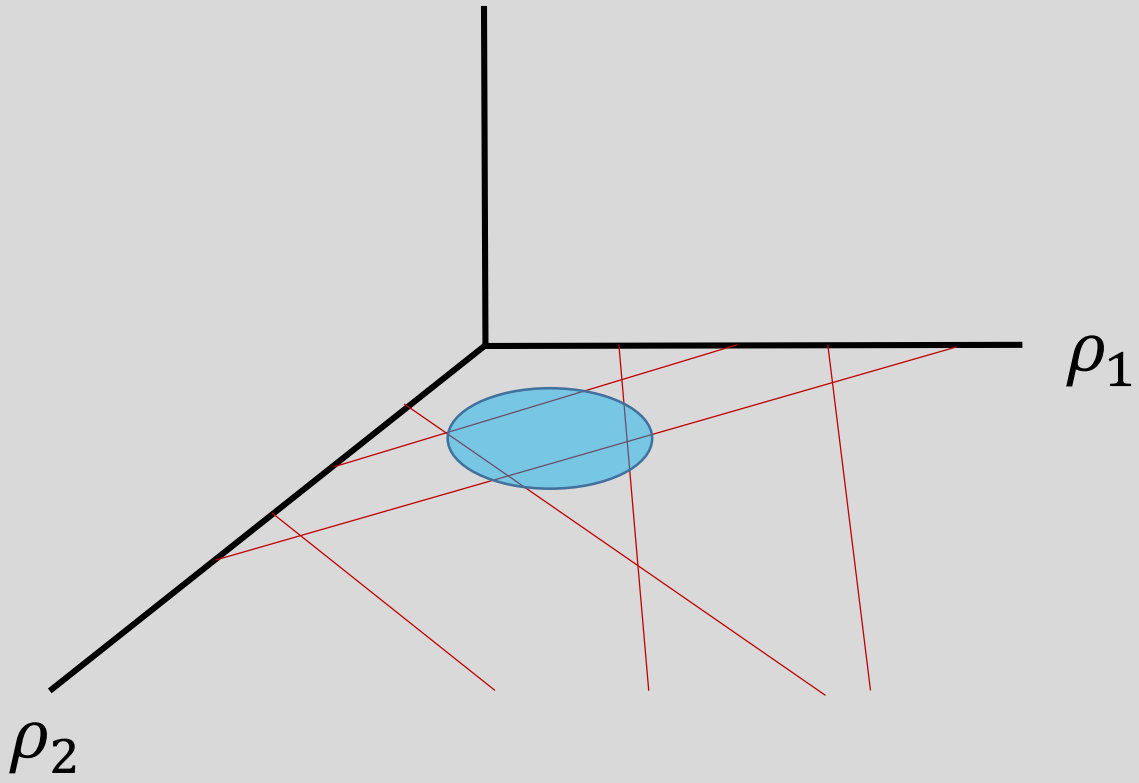
How dispersed is $U_S(\boldsymbol{\rho})$?

Let $f_i(v)$ be the density of the distribution over the "typical" buyer's value for item i.

Suppose there are m items and $\max_{i,v} f_i(v) \leq \kappa$.

With probability at least $1-\gamma$, $U_S(\boldsymbol{\rho})$ is (w,k)-dispersed with $w = \frac{\gamma}{|S|\kappa\sqrt{m}}$ and $k = m$.

$U_S(\boldsymbol{\rho})$ = average revenue



Let $f_i(v)$ be the density of the distribution over the "typical" buyer's value for item i.

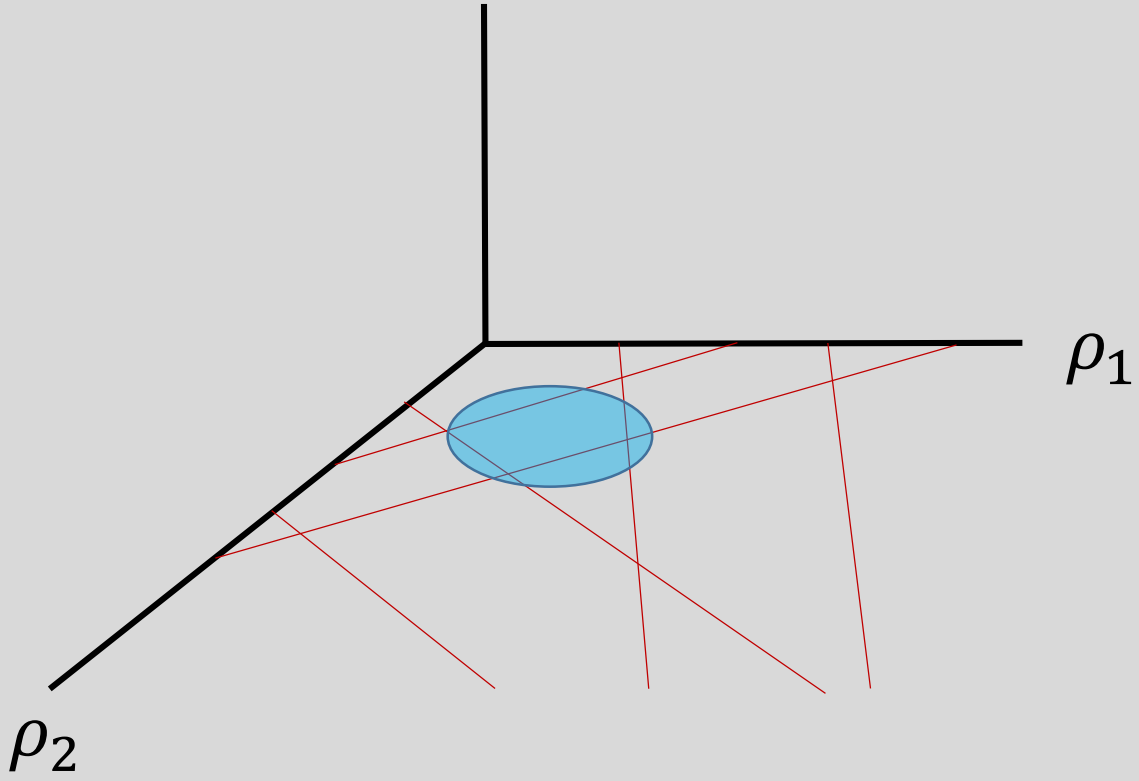
Suppose there are m items and $\max_{i,v} f_i(v) \leq \kappa$.

Let $\widehat{\boldsymbol{\rho}} \in \mathbb{R}^m$ be the price vector output by our DP algorithm.

With probability $1 - \gamma$,

$$U_S(\widehat{\boldsymbol{\rho}}) \geq \max U_S(\boldsymbol{\rho}) - \tilde{O}\left(\frac{Hm}{|S|\varepsilon} + \frac{\gamma}{|S|\kappa\sqrt{m}}\right)$$

$U_S(\boldsymbol{\rho})$ = average revenue



We also show that our DP algorithm has strong utility guarantees for many **mechanism design problems**.

Multiple bidders

Buyers with unit-demand valuations

Buyers with general valuations

2nd price auctions with reserve prices

- Introduction
- Setup
  - Overview: Algorithm configuration and auction design
  - Differential privacy
  - Private pricing design
- The algorithm
  - Privacy guarantees
  - Utility guarantees
    - Example: private multi-item pricing
- ➡ Summary

We provide a **private** algorithm for maximizing data-dependent piecewise Lipschitz functions.

Utility function	**Parameter Dimension**	**Privacy guarantee**	**Efficiency**
Piecewise concave and Lipschitz	Multiple	(ε, δ)	✓
Piecewise Lipschitz	Single	$(\varepsilon, 0)$	✓
Piecewise Lipschitz	Multiple	$(\varepsilon, 0)$	✗

Questions?