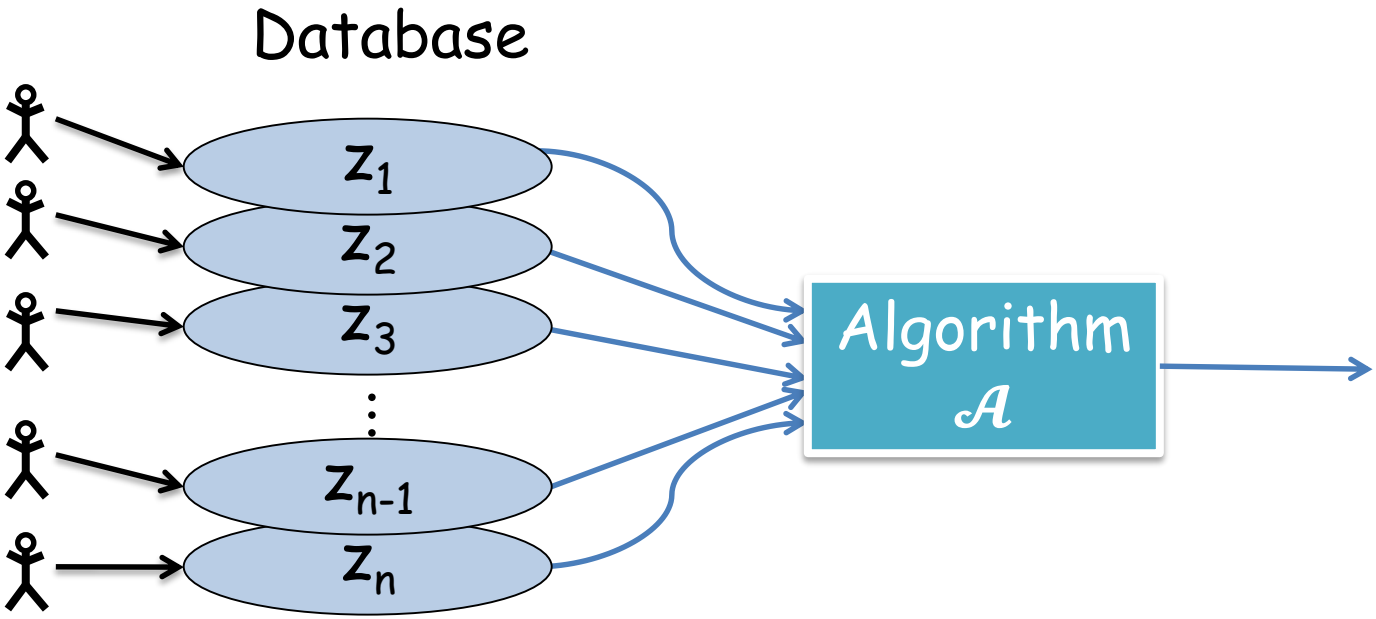
Learning Privately with Labeled and Unlabeled Examples

Uri Stemmer (BGU)
With Amos Beimel and Kobbi Nissim

Why Private Learners?

Common scenario: We want to analyze a database containing (sensitive) individual information.



Important to protect the privacy of those individuals

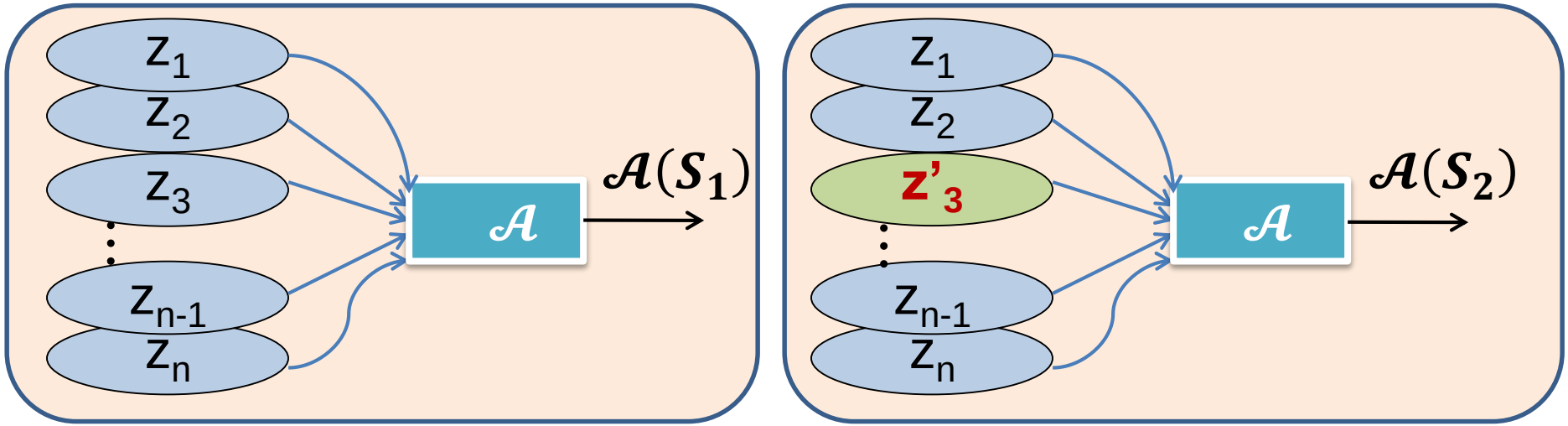
This Talk

- Definitions
  - Differential privacy
  - PAC learning model
- Previous works
- New Results
  - Decreasing the **labeled** sample complexity of private learners

Differential Privacy

Dwork, McSherry, Nissim, Smith 2006

Changing one record does not change the output distribution **"too much"**



Differential Privacy

Dwork, McSherry, Nissim, Smith 2006

Changing one record does not change the output distribution **"too much"**

A (rand) algorithm $\mathcal{A}$ is differentially private if for all neighboring databases S_1, S_2 and for all sets of outputs F:

$$\Pr[\mathcal{A}(S_1) \in F] \approx \Pr[\mathcal{A}(S_2) \in F]$$

Pure Differential Privacy

Dwork, McSherry, Nissim, Smith 2006

Changing one record does not change the output distribution **"too much"**

A (rand) algorithm $\mathcal{A}$ is ϵ differentially private if for all neighboring databases S_1, S_2 and for all sets of outputs F:

$$\Pr[\mathcal{A}(S_1) \in F] \leq e^{\epsilon} \cdot \Pr[\mathcal{A}(S_2) \in F]$$

Approx. Differential Privacy

Dwork, McSherry, Nissim, Smith 2006

Dwork, Kenthapadi, McSherry, Mironov, Naor 2006

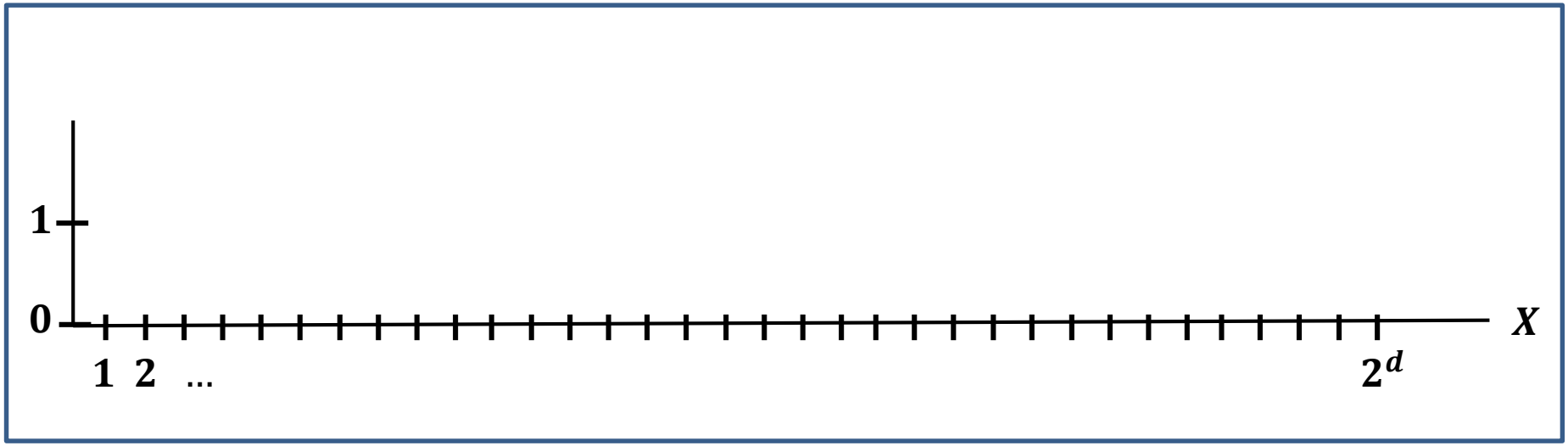
Changing one record does not change the output distribution "**too much**"

A (rand) algorithm $\mathcal{A}$ is (ϵ, δ) differentially private if for all neighboring databases S_1, S_2 and for all sets of outputs F:

$$\Pr[\mathcal{A}(S_1) \in F] \leq e^{\epsilon} \cdot \Pr[\mathcal{A}(S_2) \in F] + \delta$$

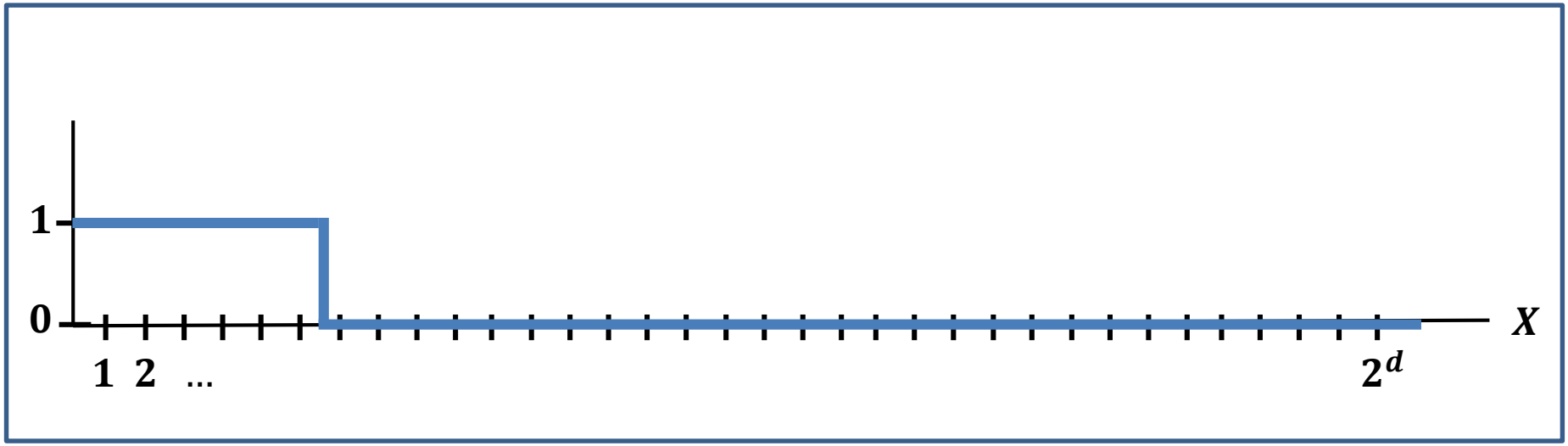
"PAC" Model [Valiant 84]

- Domain X.



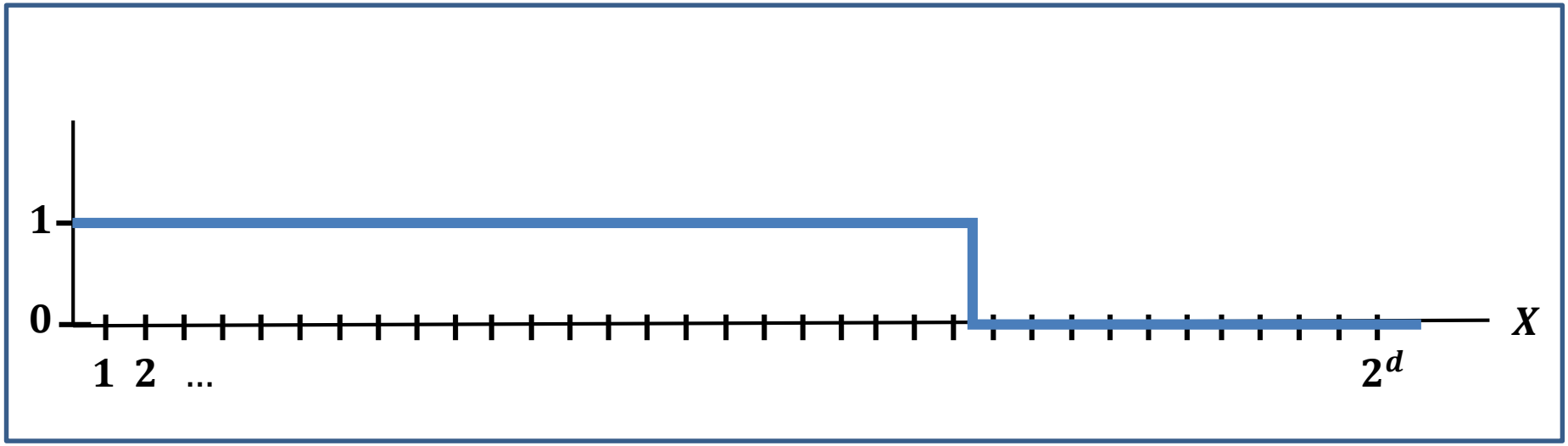
"PAC" Model [Valiant 84]

- Domain X.
- Set $\mathcal{C}$ of boolean functions over X.
  - for example: $\mathbf{INTERVAL}_d$



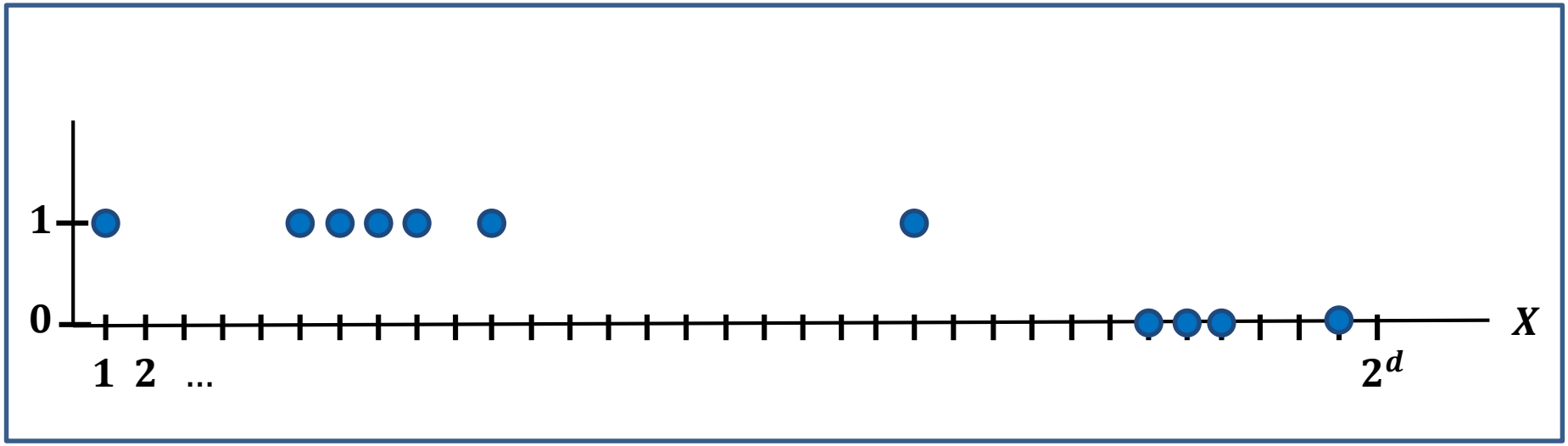
"PAC" Model [Valiant 84]

- Domain X.
- Set $\mathcal{C}$ of boolean functions over X.
  - for example: $\mathbf{INTERVAL}_d$



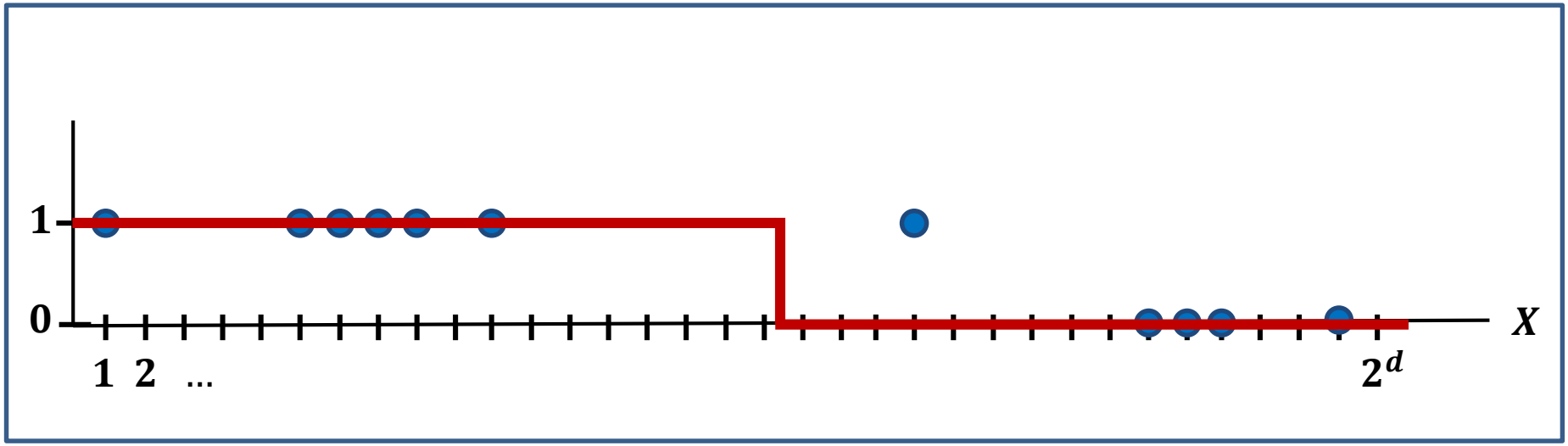
"PAC" Model [Valiant 84]

- Domain X.
- Set $\mathcal{C}$ of boolean functions over X.
  - for example: $\mathbf{INTERVAL}_d$
- Labeled sample.



"PAC" Model [Valiant 84]

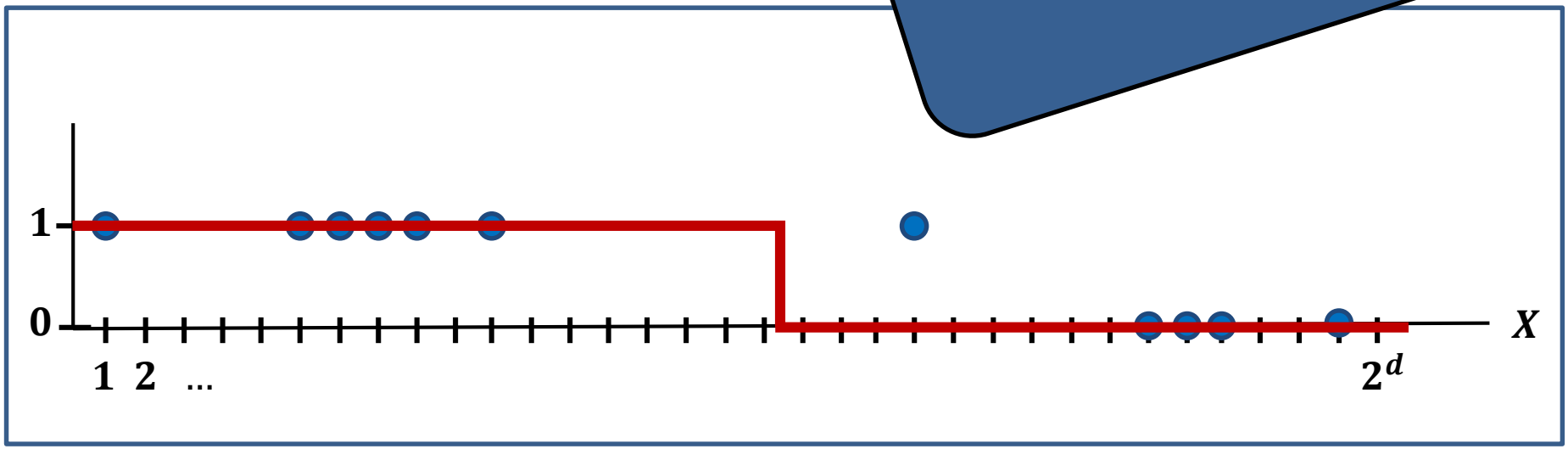
- Domain X.
- Set $\mathcal{C}$ of boolean functions over X.
  - for example: $\mathbf{INTERVAL}_d$
- Labeled sample.
- Output a classifier h.



"PAC" Model [Valiant 84]

- Domain X.
- Set $\mathcal{C}$ of boolean functions over X.
  - for example: $\mathbf{INTERVAL}_d$
- Labeled sample.
- Output a classifier h.

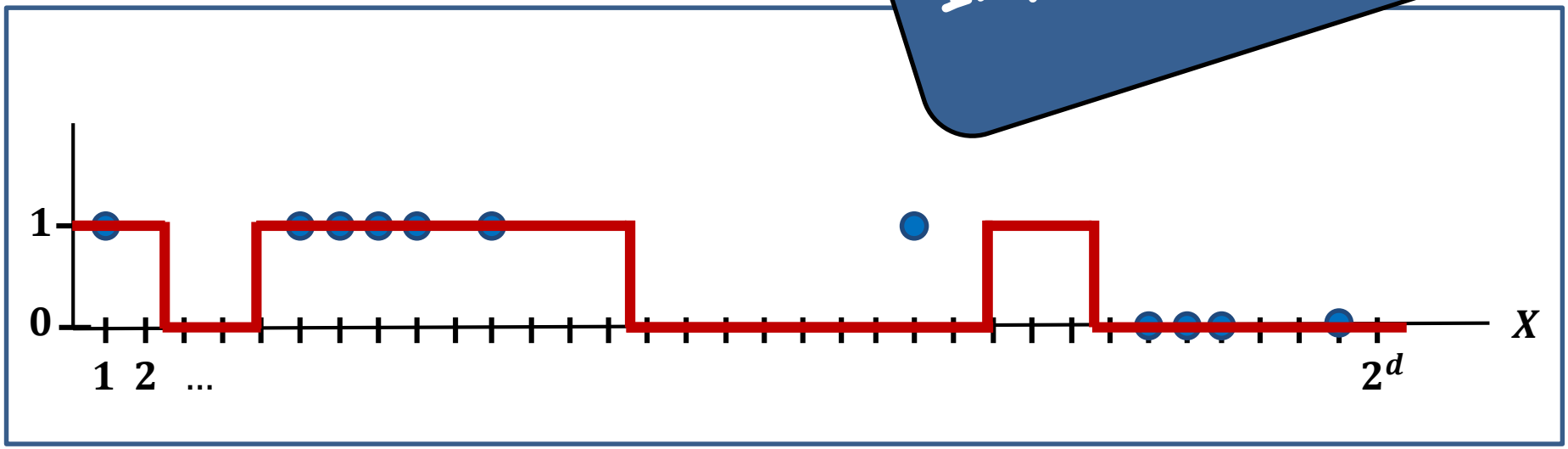
Proper if $h \in \mathcal{C}$



"PAC" Model [Valiant 84]

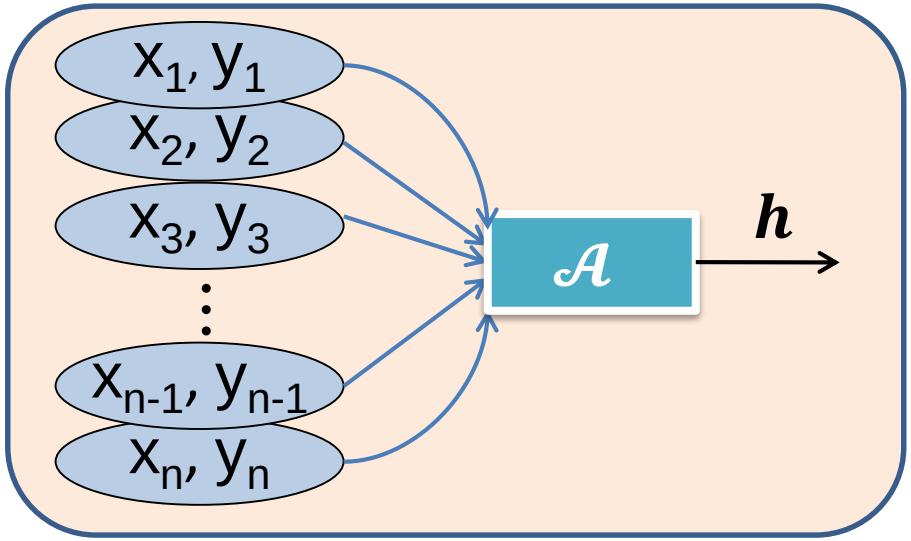
- Domain X.
- Set $\mathcal{C}$ of boolean functions over X.
  - for example: $\mathbf{INTERVAL}_d$
- Labeled sample.
- Output a classifier h.

Improper if $h \notin \mathcal{C}$



PAC learner

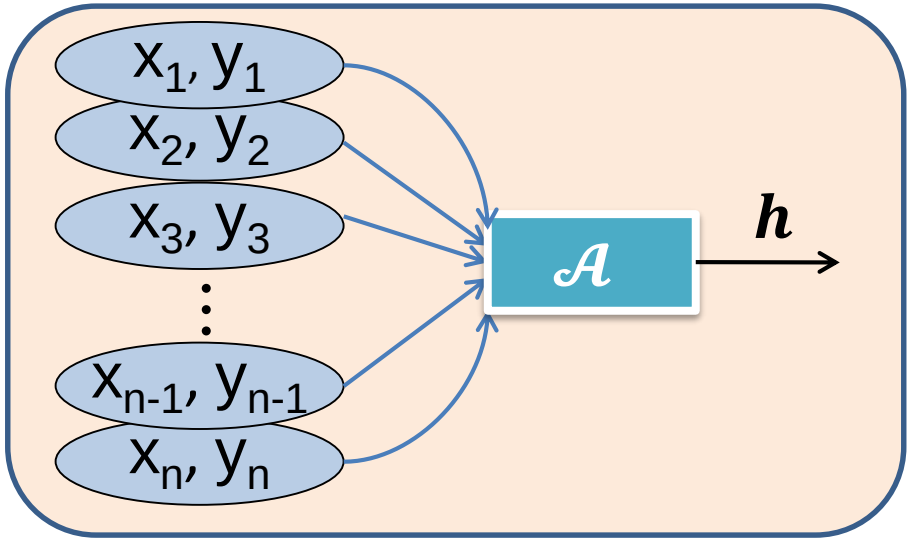
$\mathcal{A}$ is a PAC learner for $\mathcal{C}$.



Private PAC learner

Kasiviswanathan, Lee, Nissim, Raskhodnikova, Smith 08

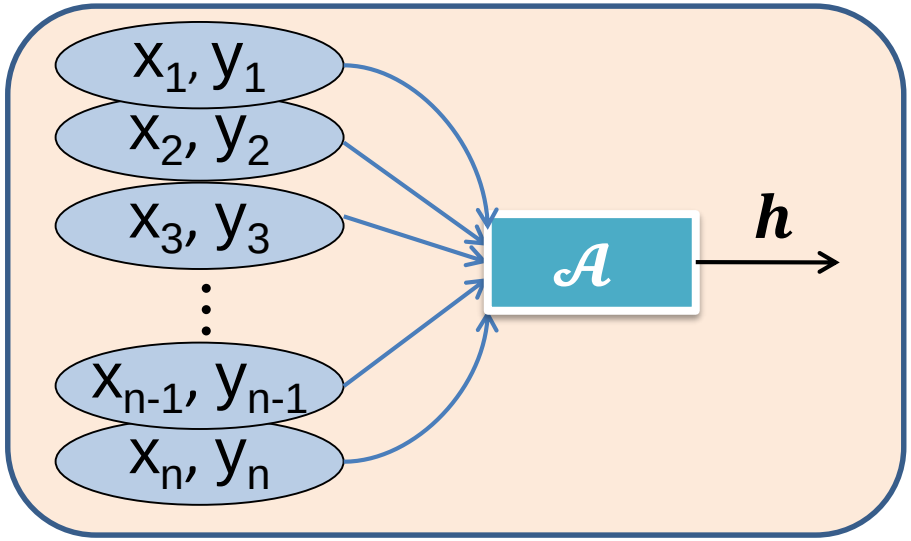
- **Learning:** $\mathcal{A}$ is a PAC learner for $\mathcal{C}$.
- **Privacy:** $\mathcal{A}$ preserves differential privacy.
  - Label, sample, presence in database may all be sensitive!



Private PAC learner

Kasiviswanathan, Lee, Nissim, Raskhodnikova, Smith 08

- **Learning:** $\mathcal{A}$ is a PAC learner for $\mathcal{C}$.
- **Privacy:** $\mathcal{A}$ preserves differential privacy.
  - Label, sample, presence in database may all be sensitive!



The Big Question: What is the sample complexity of private learning?

Previous Works:

Previous Work:
Sample Complexity Bounds

[VC 71, BEHW 89]: Every concept class $\mathcal{C}$ can be learned non-privately using $\boldsymbol{O(\mathrm{VC}(\mathcal{C}))}$ samples.

[KLNRS 08]: Every finite concept class $\mathcal{C}$ can be learned privately using $\boldsymbol{O(\log|\mathcal{C}|)}$ samples.

Previous Work:
Sample Complexity Bounds

[VC 71, BEHW 89]: Every concept class $\mathcal{C}$ can be learned non-privately using $\boldsymbol{O(\mathrm{VC}(\mathcal{C}))}$ samples.

[KLNRS 08]: Every finite concept class $\mathcal{C}$ can be learned privately using $\boldsymbol{O(\log|\mathcal{C}|)}$ samples.

[BKN 10]: $\boldsymbol{\Omega(\log|\mathcal{C}|)}$ samples are (generally) needed for pure-private proper-learning.

Previous Work:
Sample Complexity Bounds

[VC 71, BEHW 89]: Every concept class $\mathcal{C}$ can be learned non-privately using $O(\mathrm{VC}(\mathcal{C}))$ samples.

[KLNRS 08]: Every finite concept class $\mathcal{C}$ can be learned privately using $O(\log|\mathcal{C}|)$ samples.

[BKN 10]: $\Omega(\log|\mathcal{C}|)$ samples are (generally) needed for pure-private proper-learning.

Previous Work:
Improper Learners

[FX 14]: Every pure-private improper-learner for INTERVAL_d must use $\boldsymbol{\Omega(d)}$ samples.

- Strong separation from the $O(1)$ non-private sample complexity

Previous Work:

Improper Learners

[FX 14]: Every pure-private improper-learner for INTERVAL_d must use $\Omega(d)$ samples.

- Strong separation from the $O(1)$ non-private sample complexity

Approx. Private Learners

[BNS 13]: Approx-private proper-learner for INTERVAL_d using $2^{O(\log^* d)}$ samples.

Previous Work:

Improper Learners

[FX 14]: Every pure-private improper-learner for INTERVAL_d must use $\Omega(d)$ samples.

- Strong separation from the $O(1)$ non-private sample complexity

Approx. Private Learners

[BNS 13]: Approx-private proper-learner for INTERVAL_d using $2^{O(\log^* d)}$ samples.

[BNSV 14]: Every approx-private proper-learner for INTERVAL_d must use $\Omega(\log^* d)$ samples.

Our Contribution:

Split the Labeled and the Unlabeled Sample Complexities

Another approach for reducing the sample complexity

- In many cases: **Unlabeled** data is easy to come by
  Labeled data is expensive

- Inspired by the work of [BF 13] on the Active Learning model

Results

- **Generic construction:**
  Every finite concept class C can be learned privately using $O(\mathrm{VC}(C))$ labeled examples.
  - The construction requires poly many unlabeled examples.

Results

- **Generic construction:**
  Every finite concept class C can be learned privately using $O(\mathrm{VC}(C))$ labeled examples.
  - The construction requires poly many unlabeled examples.

- **Boosting the labeled sample complexity:**
  Given a private learner for a concept class C, it is possible to reduce its labeled sample complexity to $O(\mathrm{VC}(C))$.
  - While maintaining the unlabeled sample complexity.

Results

- **Generic construction:**
  Every finite concept class C can be learned privately using $O(\mathrm{VC}(C))$ labeled examples.
  - The construction requires poly many unlabeled examples.

- **Boosting the labeled sample complexity:**
  Given a private learner for a concept class C, it is possible to reduce its labeled sample complexity to $O(\mathrm{VC}(C))$.
  - While maintaining the unlabeled sample complexity.

- **Active Learning:**
  - Not in this talk.

Results

- **Generic construction:**
  Every finite concept class C can be learned privately using $O(\mathrm{VC}(C))$ labeled examples.
  - The construction requires poly many unlabeled examples.

- **Boosting the labeled sample complexity:**
  Given a private learner for a concept class C, it is possible to reduce its labeled sample complexity to $O(\mathrm{VC}(C))$.
  - While maintaining the unlabeled sample complexity.

- **Active Learning:**
  - Not in this talk.

Algorithm LabelBoost

Goal: Reduce the **labeled** sample complexity of a given learner $\mathcal{A}$.

Inputs:
- Base learner $\mathcal{A}$ with sample complexity n.
- Database S of size n, only partially labeled.

Algorithm LabelBoost

Goal: Reduce the **labeled** sample complexity of a given learner $\mathcal{A}$.

Inputs:
- Base learner $\mathcal{A}$ with sample complexity n.
- Database S of size n, only partially labeled.

1. Let H be the set of all dichotomies over S realized by the target concept class C.

Algorithm LabelBoost

Goal: Reduce the labeled sample complexity of a given learner $\mathcal{A}$.

Inputs:
- Base learner $\mathcal{A}$ with sample complexity n.
- Database S of size n, only partially labeled.

1. Let H be the set of all dichotomies over S realized by the target concept class C.

2. Choose $h \in H$ using the exponential mechanism with the **labeled portion** of S.

3. Relabel S using h, and execute $\mathcal{A}$.

Algorithm LabelBoost

Goal: Reduce the labeled sample complexity of a given learner $\mathcal{A}$.

Inputs:
- Base learner $\mathcal{A}$ with sample complexity n.
- Database S of size n, only partially labeled.

x_1 , y_1
x_2 , y_2
x_3 , y_3
x_4 , y_4
⋮
x_t , y_t
x_{t+1} , ?
x_{t+2} , ?
x_{t+3} , ?
⋮
x_n , ?

1. Let H be the set of all dichotomies over S realized by the target concept class C.

2. Choose $h \in H$ using the exponential mechanism with the **labeled portion** of S.

3. Relabel S using h, and execute $\mathcal{A}$.

Algorithm LabelBoost

Goal: Reduce the labeled sample complexity of a given learner $\mathcal{A}$.

Inputs:
- Base learner $\mathcal{A}$ with sample complexity n.
- Database S of size n, only partially labeled.

x_1 , y_1
x_2 , ~~y_2~~ $\hat{y}_2$
x_3 , y_3
x_4 , y_4
⋮
x_t , y_t
x_{t+1} , ~~?~~ $\hat{y}_{t+1}$
x_{t+2} , ~~?~~ $\hat{y}_{t+2}$
x_{t+3} , ~~?~~ $\hat{y}_{t+3}$
⋮
x_n , ~~?~~ $\hat{y}_n$

1. Let H be the set of all dichotomies over S realized by the target concept class C.

2. Choose $h \in H$ using the exponential mechanism with the **labeled portion** of S.

3. Relabel S using h, and execute $\mathcal{A}$.

Algorithm LabelBoost

Goal: Reduce the **labeled** sample complexity of a given learner $\mathcal{A}$.

Inputs:
- Base learner $\mathcal{A}$ with sample complexity n.
- Database S of size n, only partially labeled.

Utility

1. Let H be the set of all dichotomies over S realized by the target concept class C.

2. Choose $h \in H$ using the exponential mechanism with the **labeled portion** of S.

3. Relabel S using h, and execute $\mathcal{A}$.

Algorithm LabelBoost

Goal: Reduce the labeled sample complexity of a given learner $\mathcal{A}$.

Inputs:
- Base learner $\mathcal{A}$ with sample complexity n.
- Database S of size n, only partially labeled.

Utility

$\exists f \in H$ s.t. $\mathbf{error}_S(f) = 0$

1. Let H be the set of all dichotomies over S realized by the target concept class C.

2. Choose $h \in H$ using the exponential mechanism with the **labeled portion** of S.

3. Relabel S using h, and execute $\mathcal{A}$.

Algorithm LabelBoost

Goal: Reduce the labeled sample complexity of a given learner $\mathcal{A}$.

Inputs:
- Base learner $\mathcal{A}$ with sample complexity n.
- Database S of size n, only partially labeled.

1. Let H be the set of all dichotomies over S realized by the target concept class C.
2. Choose $h \in H$ using the exponential mechanism with the **labeled portion** of S.
3. Relabel S using h, and execute $\mathcal{A}$.

Utility

$\exists f \in H$ s.t. $\mathbf{error}_S(f) = 0$

If S contains $\approx \log|S|$ labels then h is close to the target concept

Algorithm LabelBoost

Goal: Reduce the labeled sample complexity of a given learner $\mathcal{A}$.

Inputs:
- Base learner $\mathcal{A}$ with sample complexity n.
- Database S of size n, only partially labeled.

1. Let H be the set of all dichotomies over S realized by the target concept class C.
2. Choose $h \in H$ using the exponential mechanism with the **labeled portion** of S.
3. Relabel S using h, and execute $\mathcal{A}$.

Utility

$\exists f \in H$ s.t. $\mathbf{error}_S(f) = 0$

If S contains $\approx \log|S|$ labels then h is close to the target concept

$\mathcal{A}$ returns a hypothesis that is close to h

Algorithm LabelBoost

Goal: Reduce the labeled sample complexity of a given learner $\mathcal{A}$.

Inputs:
- Base learner $\mathcal{A}$ with sample complexity n.
- Database S of size n, only partially labeled.

Privacy

1. Let H be the set of all dichotomies over S realized by the target concept class C.

2. Choose $h \in H$ using the exponential mechanism with the **labeled portion** of S.

3. Relabel S using h, and execute $\mathcal{A}$.

Algorithm LabelBoost

Goal: Reduce the labeled sample complexity of a given learner $\mathcal{A}$.

Inputs:
- Base learner $\mathcal{A}$ with sample complexity n.
- Database S of size n, only partially labeled.

1. Let H be the set of all dichotomies over S realized by the target concept class C.
2. Choose $h \in H$ using the exponential mechanism with the **labeled portion** of S.
3. Relabel S using h, and execute $\mathcal{A}$.

Privacy

Difficulty: The set H strongly depends on the input points, so outputting an $h \in H$ may breach privacy.

Algorithm LabelBoost

Goal: Reduce the labeled sample complexity of a given learner $\mathcal{A}$.

Inputs:
- Base learner $\mathcal{A}$ with sample complexity n.
- Database S of size n, only partially labeled.

1. Let H be the set of all dichotomies over S realized by the target concept class C.
2. Choose $h \in H$ using the exponential mechanism with the labeled portion of S.
3. Relabel S using h, and execute $\mathcal{A}$.

Privacy

Difficulty: The set H strongly depends on the input points, so outputting an $h \in H$ may breach privacy.

Idea: Analyze the distribution of the relabeled databases

Algorithm LabelBoost

Goal: Reduce the labeled sample complexity of a given learner $\mathcal{A}$.

Inputs:
- Base learner $\mathcal{A}$ with sample complexity n.
- Database S of size n, only partially labeled.

1. Let H be the set of all dichotomies over S realized by the target concept class C.
2. Choose $h \in H$ using the exponential mechanism with the labeled portion of S.
3. Relabel S using h, and execute $\mathcal{A}$.

Privacy

Difficulty: The set H strongly depends on the input points, so outputting an $h \in H$ may breach privacy.

Idea: Analyze the distribution of the relabeled databases

$\Rightarrow$ Reduces the labeled sample complexity logarithmically!

Improving the Algorithm

We saw one step: Reducing the labeled sample complexity from n to $O(\mathbf{VC}(C) \cdot \log n)$.

Using recursion:

Reduce the labeled sample complexity to $\mathbf{VC}(C)$.

Improving the Algorithm

We saw one step: Reducing the labeled sample complexity from n to $O(\mathrm{VC}(C) \cdot \log n)$.

Using recursion:

Reduce the labeled sample complexity to $\mathrm{VC}(C)$.

Nice properties:

- If $\mathcal{A}$ is proper, then the resulting learner is proper.
- If $\mathcal{A}$ is ϵ-private, then the resulting learner is ϵ-private.
- The labeled sample complexity has no dependency in δ.
  - The dependency in ϵ can be removed in the active model.

Summary and Open Problems

What we saw:

- Given a private learner $\mathcal{A}$, it is possible to reduce its labeled sample complexity to $O(\mathbf{VC}(C))$.
- The labeled sample complexity of private learners is characterized by the VC dimension.

Open problem:

Efficient algorithms? Efficient transformation?

The sample complexity of approx. private learners?