Improved learning of k-parities

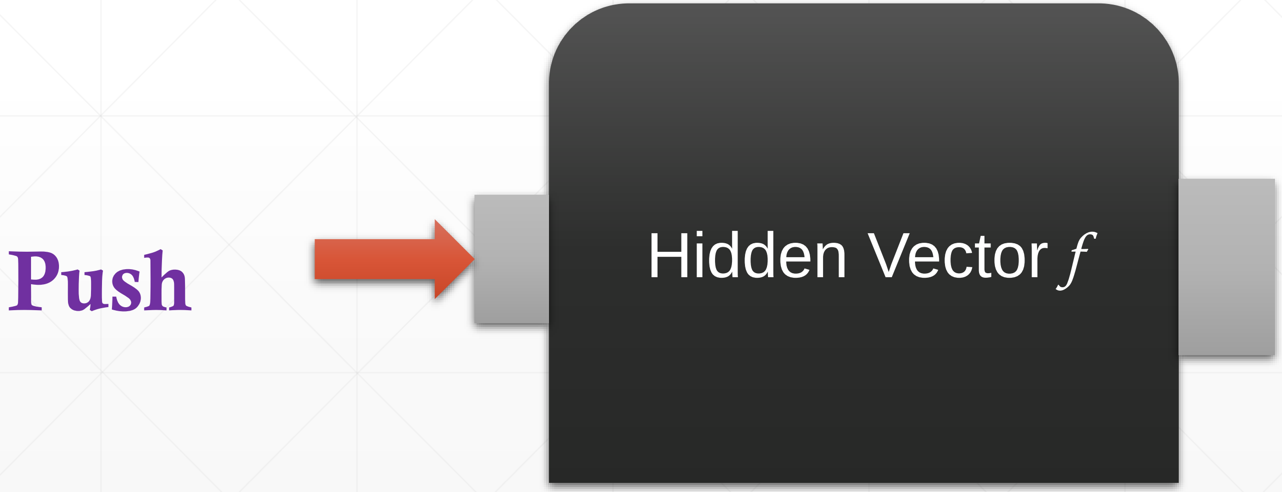
Arnab Bhattacharyya
Indian Institute of Science
India

Ameet Gadekar
Aalto University
Finland

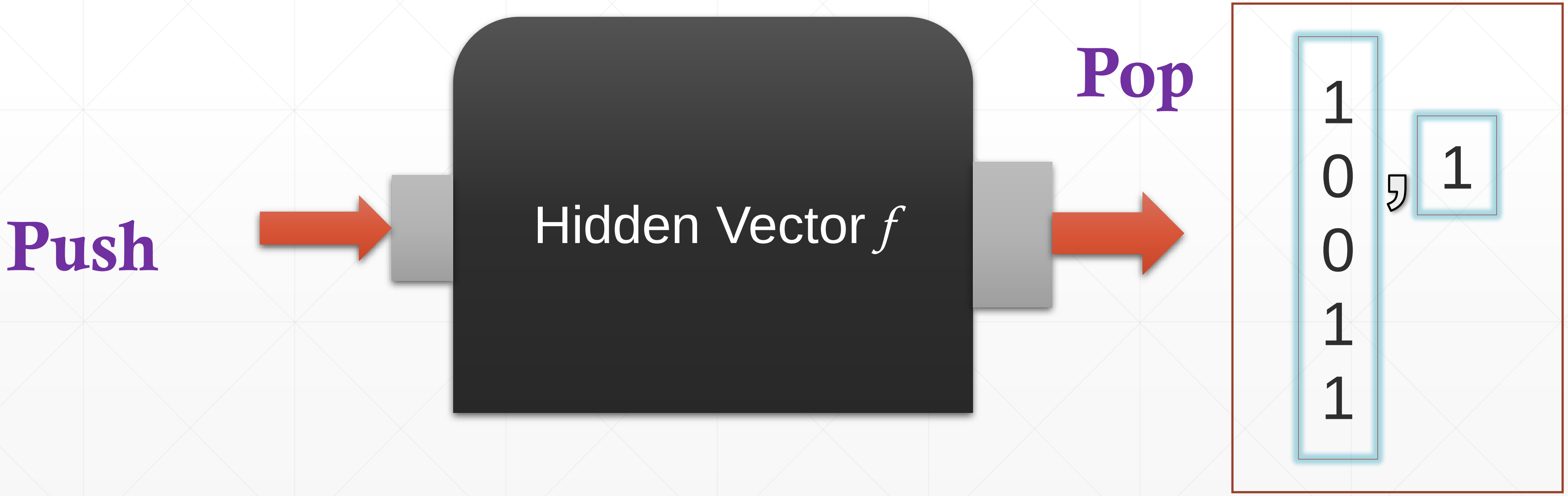
Ninad Rajgopal
University of Oxford
UK

COCOON, Qingdao, China, July 2018

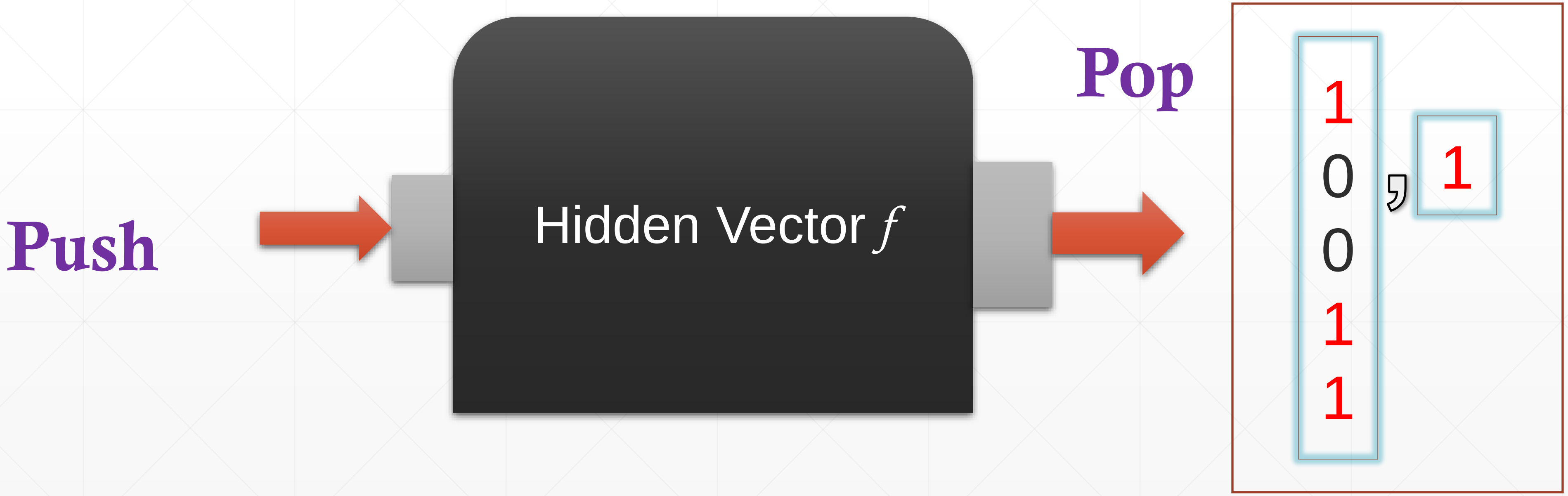
The Learning Parity Problem



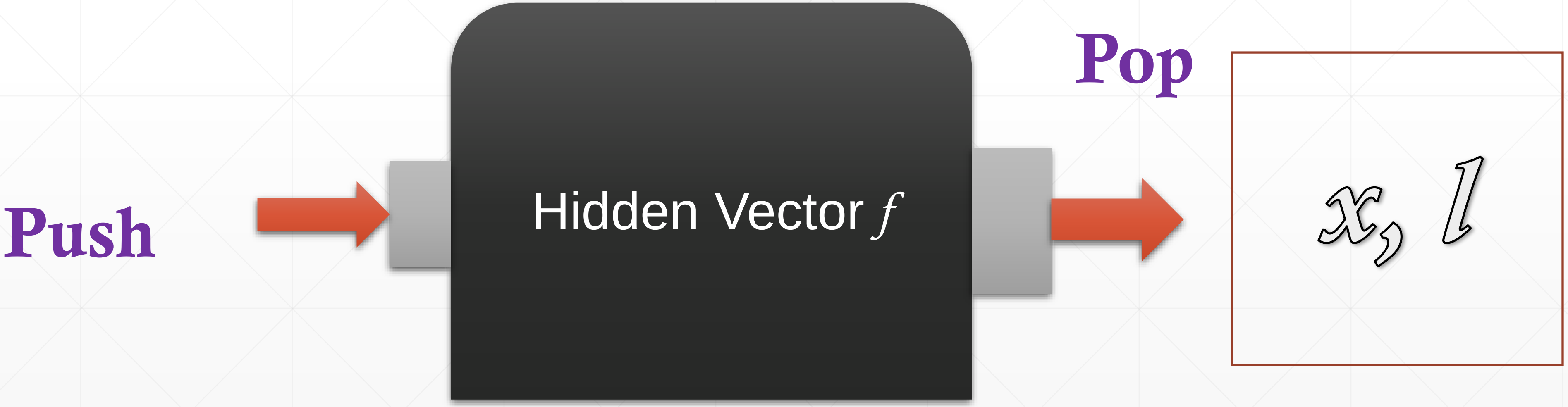
The Learning Parity Problem



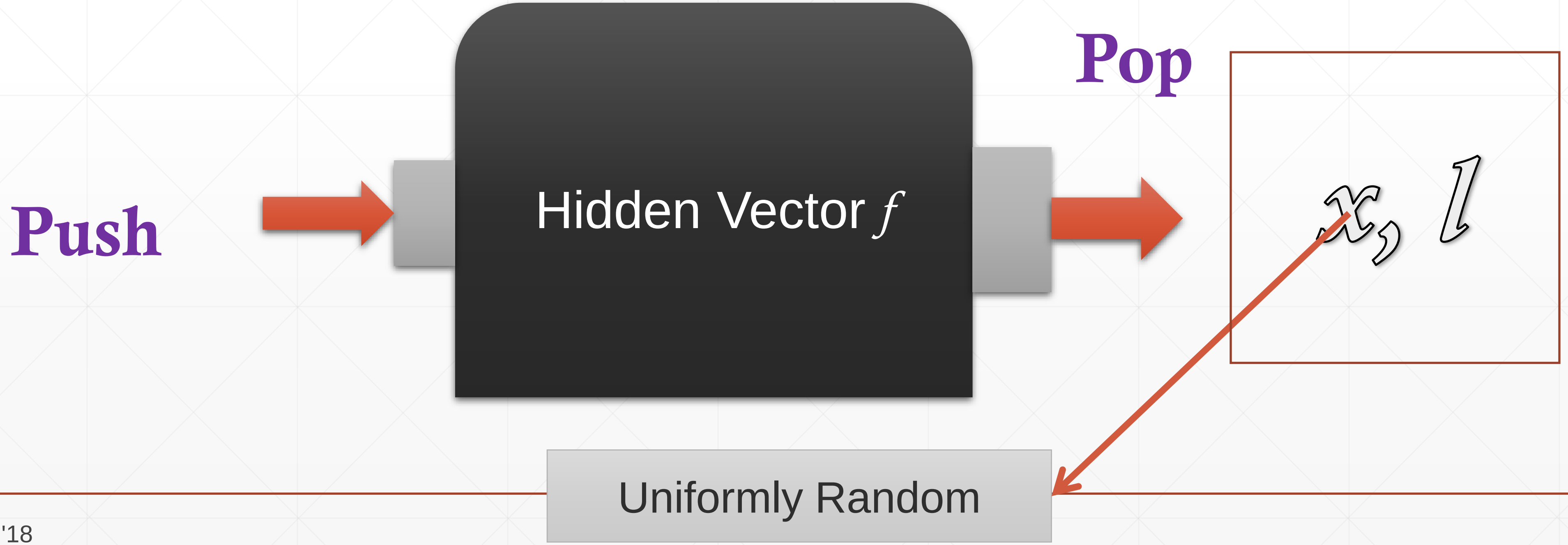
The Learning Parity Problem



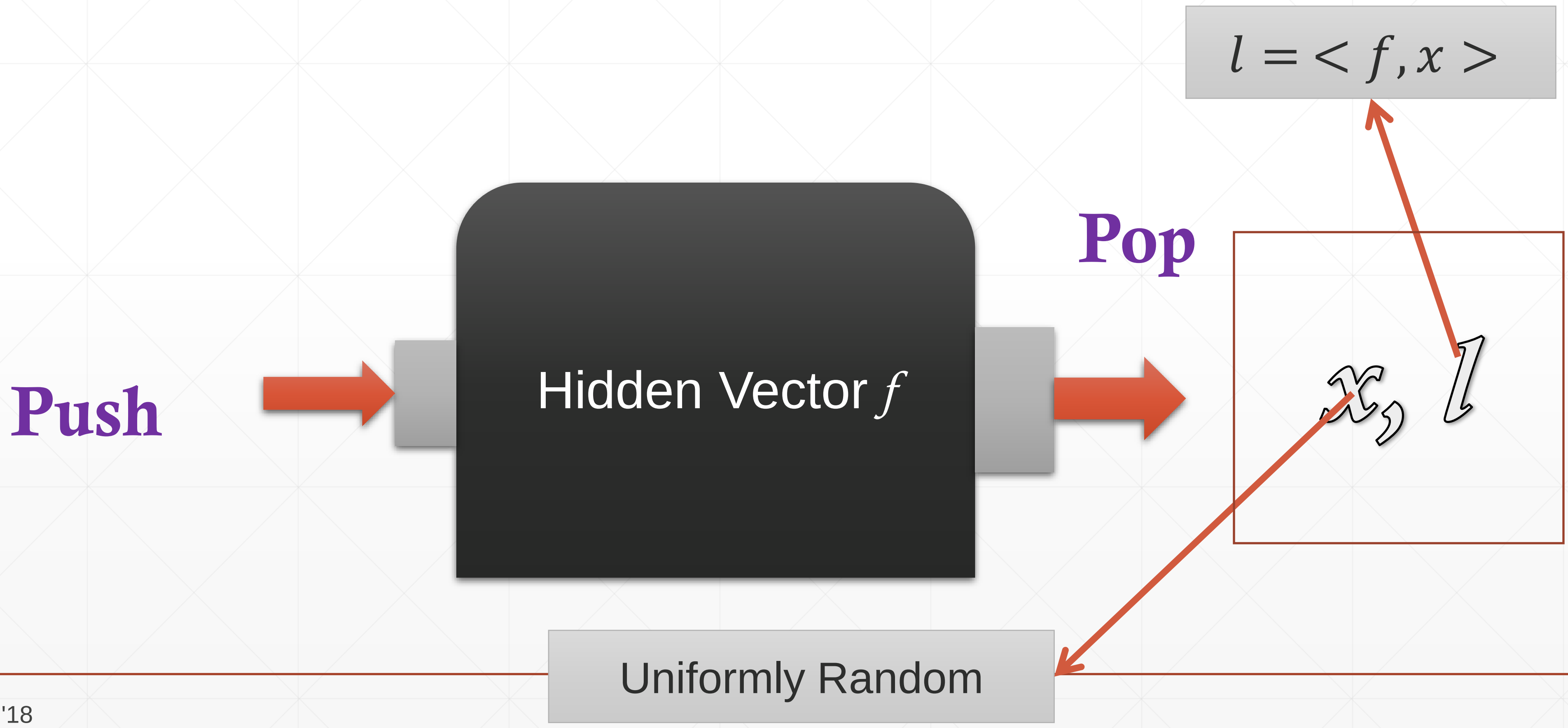
The Learning Parity Problem



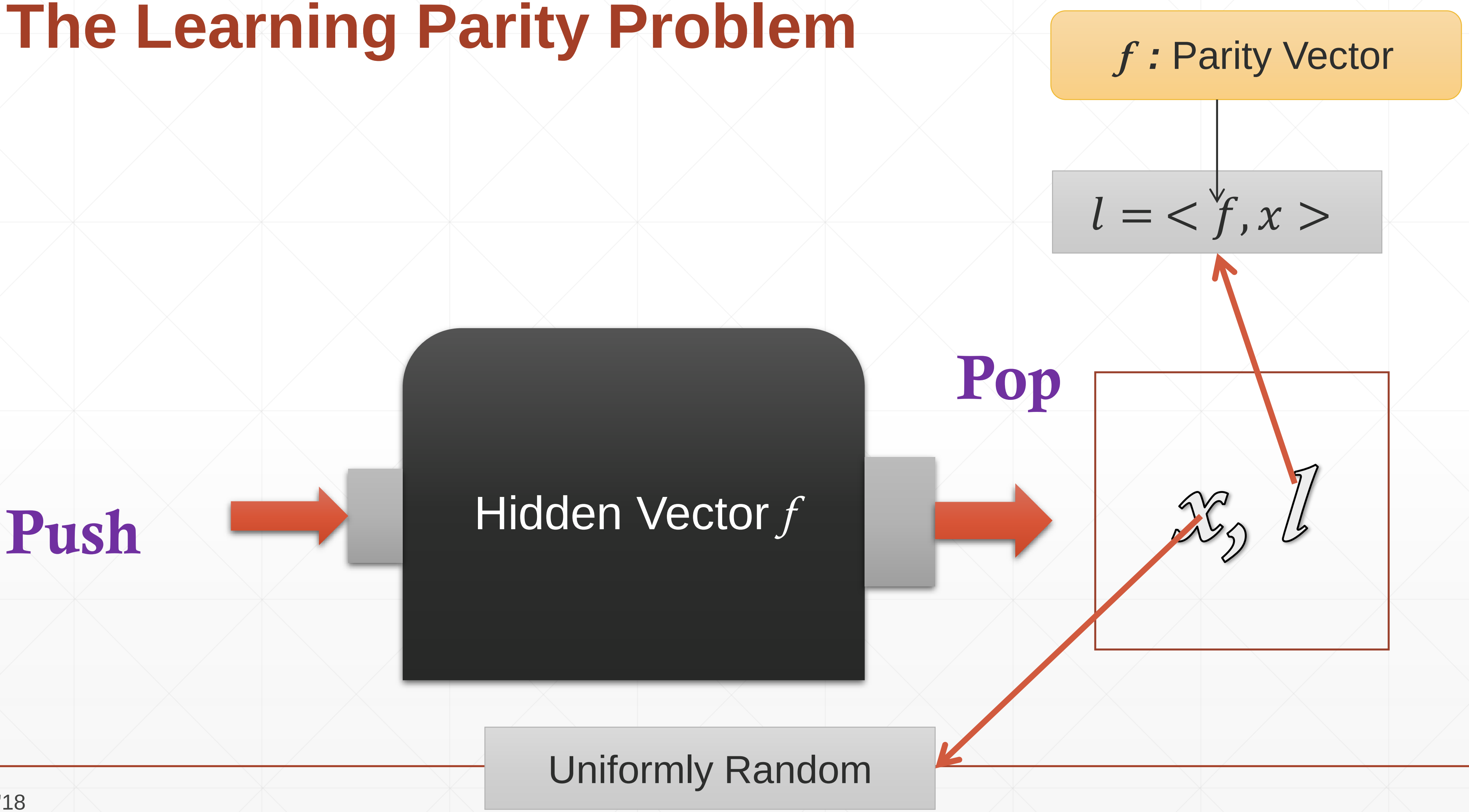
The Learning Parity Problem



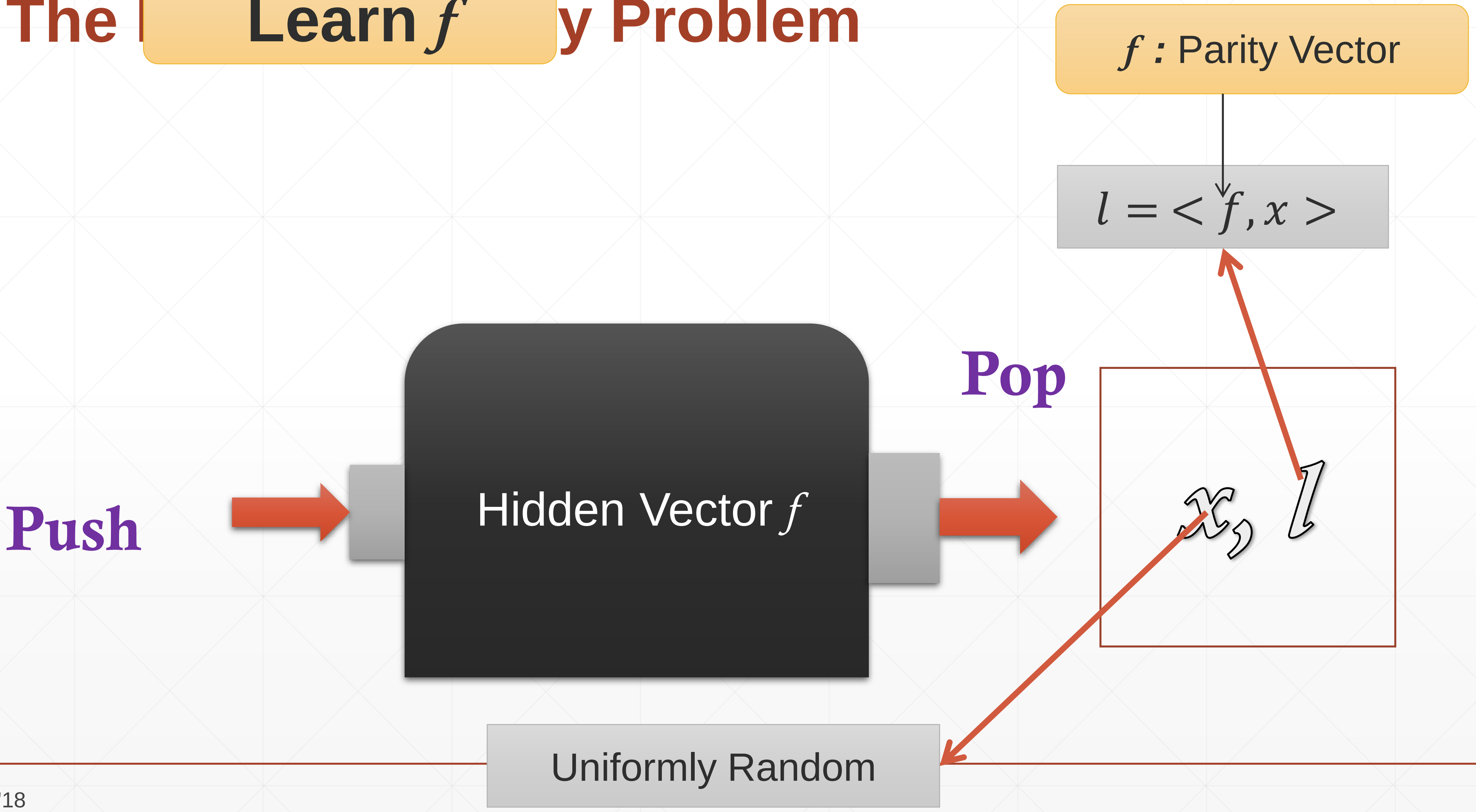
The Learning Parity Problem



The Learning Parity Problem



The Learn f y Problem



Learn f minimizing

- Number of samples
- Running time

The Learning Parity Problem

Gaussian elimination

- Uses $O(n)$ samples
- Runs in time $O(n^3)$

Learning Parity with Noise

- Same setup
- But the environment is noisy with **noise rate η**
- The labels are **flipped independently** with probability η
- Learn f **minimizing** the number of **samples** and **running time**

Learning Parity with Noise

- Can be solved using brute force algorithm that runs in time $\mathbf{O}(\mathbf{2}^{\boldsymbol{n}})$

- Best known algorithm running time $\mathbf{O}(\mathbf{2}^{\frac{\boldsymbol{n}}{\boldsymbol{log\,n}}})$ by [*Blum, Kalai, Wasserman '03*]

Learning Parities

- Central Problem in Learning theory [*Feldman et al ‘09*]
- Coding theory
- Cryptography
- Lower bounds : **Open**

Learning k-Parity

- In this paper, study the variant problem in which f is ***k*-sparse** i.e. $|f| = k$ and $k \ll n$

- *First result* - Learning k-Parity **without noise**

- *Second result* - Learning k-Parity **with noise**

Learning k-Parity without noise

Learning k-Parity without noise

- Two approaches to learn f

Learning k-Parity without noise

- Two approaches to learn f

	Gaussian Elimination	Halving Algorithm
Sample Complexity	$O(n)$	$O\left(\log\binom{n}{k}\right)$
Time Complexity	$O(n^3)$	$O(n^{\frac{k}{2}})$

Learning *k*-Parity without noise

- Current best *trade-offs* between sample complexity and running time given by (BGM) *Buhrman, Garcia-Soriano and Matsliah (2010)* in the stronger **Mistake bound model**.

- This paper - we **improve** the current *best trade offs*.

Online Mistake Bound model

- Oracle provides an **unlabeled** example x
- Learner **predicts** the label $\tilde{f}(x)$
- Oracle gives the **correct** label $f(x)$
- Learner can **update** its solution space depending upon the answer revealed

Each Round

The process repeats

Online Mistake Bound model

- **Mistake**: $f(x) \neq \tilde{f}(x)$

- *Learn* f minimizing
  - **Mistake** bound
  - Per round running **time**

- **Adversarial** model, **more difficult** than PAC model [*Blum'94*].

Our results for noiseless case

	This paper
Running time:	$e^{-k/4.01} \cdot \binom{t}{k} \cdot \tilde{O}(\left(\frac{kn}{t}\right)^2)$
Mistake bound:	$(1+o(1))\frac{kn}{t} + log\binom{t}{k}$

Our results for noiseless case

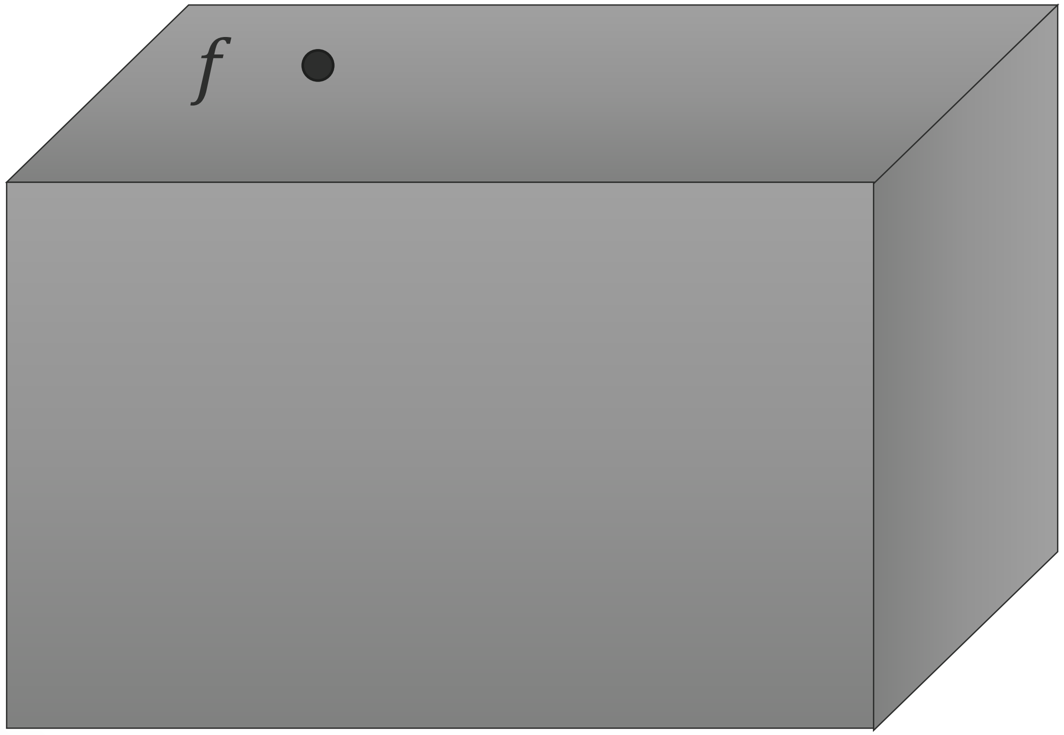
	This paper	BGM'10
Running time:	$\boxed{e^{-k/4.01}} \cdot \binom{t}{k} \cdot \tilde{O}(\left(\frac{kn}{t}\right)^2)$	$O(\binom{t}{k}\left(\frac{kn}{t}\right)^2)$
Mistake bound:	$(1+o(1))\frac{kn}{t} + log\binom{t}{k}$	$k\lceil\frac{n}{t}\rceil + \lceil log\binom{t}{k}\rceil$

Idea behind the BGM algorithm

- Two approaches to learn f

	Gaussian Elimination	Halving Algorithm
Sample Complexity	$O(n)$	$O\left(\log\binom{n}{k}\right)$
Time Complexity	$O(n^3)$	$O(n^{\frac{k}{2}})$

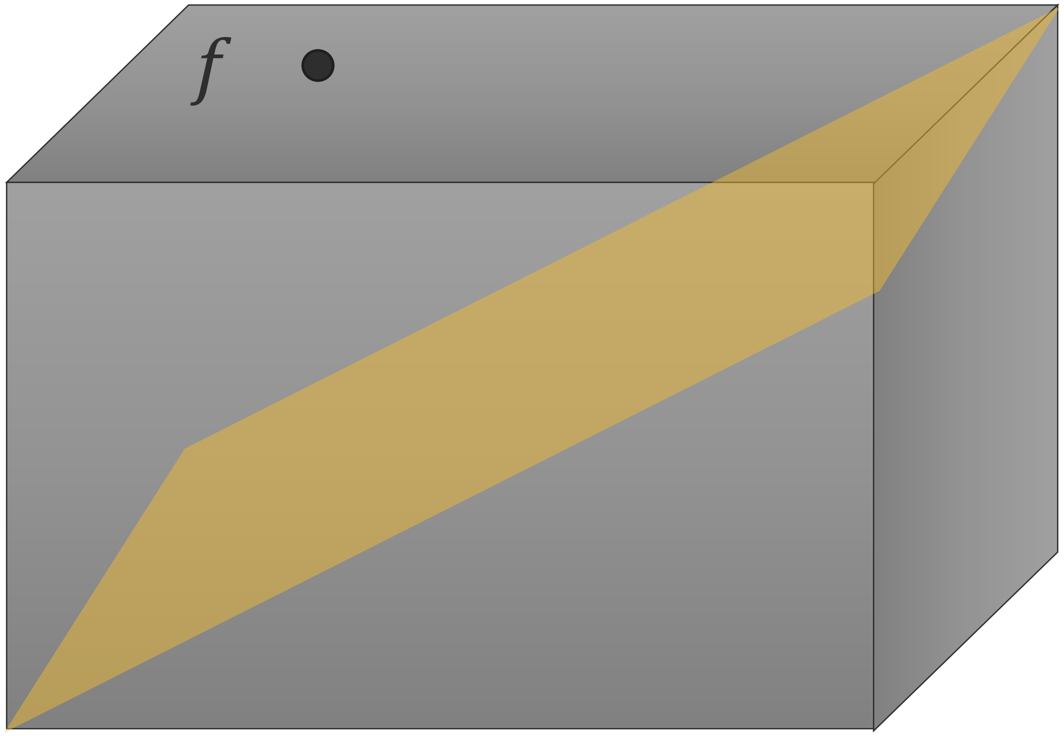
Gaussian Elimination



***Gaussian Elimination*:** Geometrically

Consider the vector space $\mathbb{F}_2^n$

Gaussian Elimination

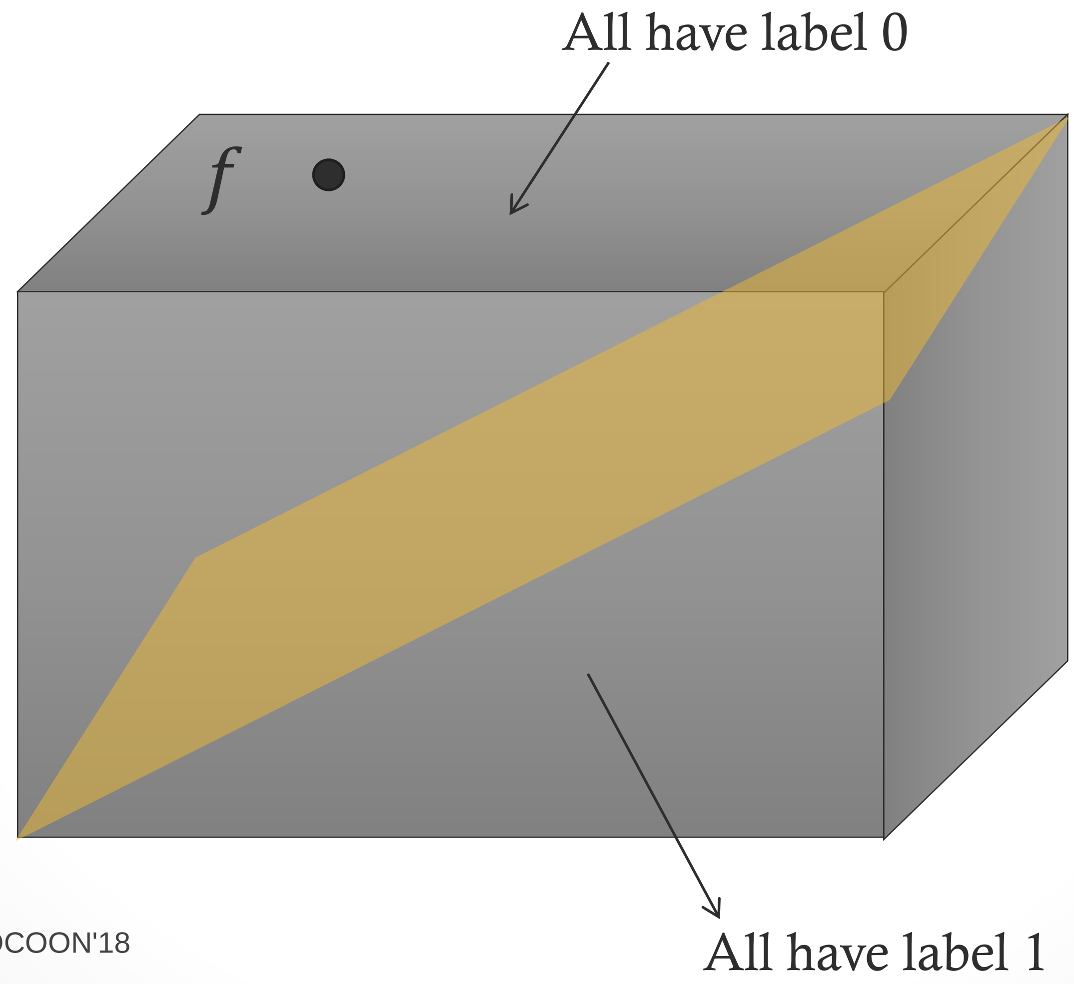


***Gaussian Elimination*:** Geometrically

Consider the vector space $\mathbb{F}_2^n$

An unlabeled example x specifies a hyper plane

Gaussian Elimination



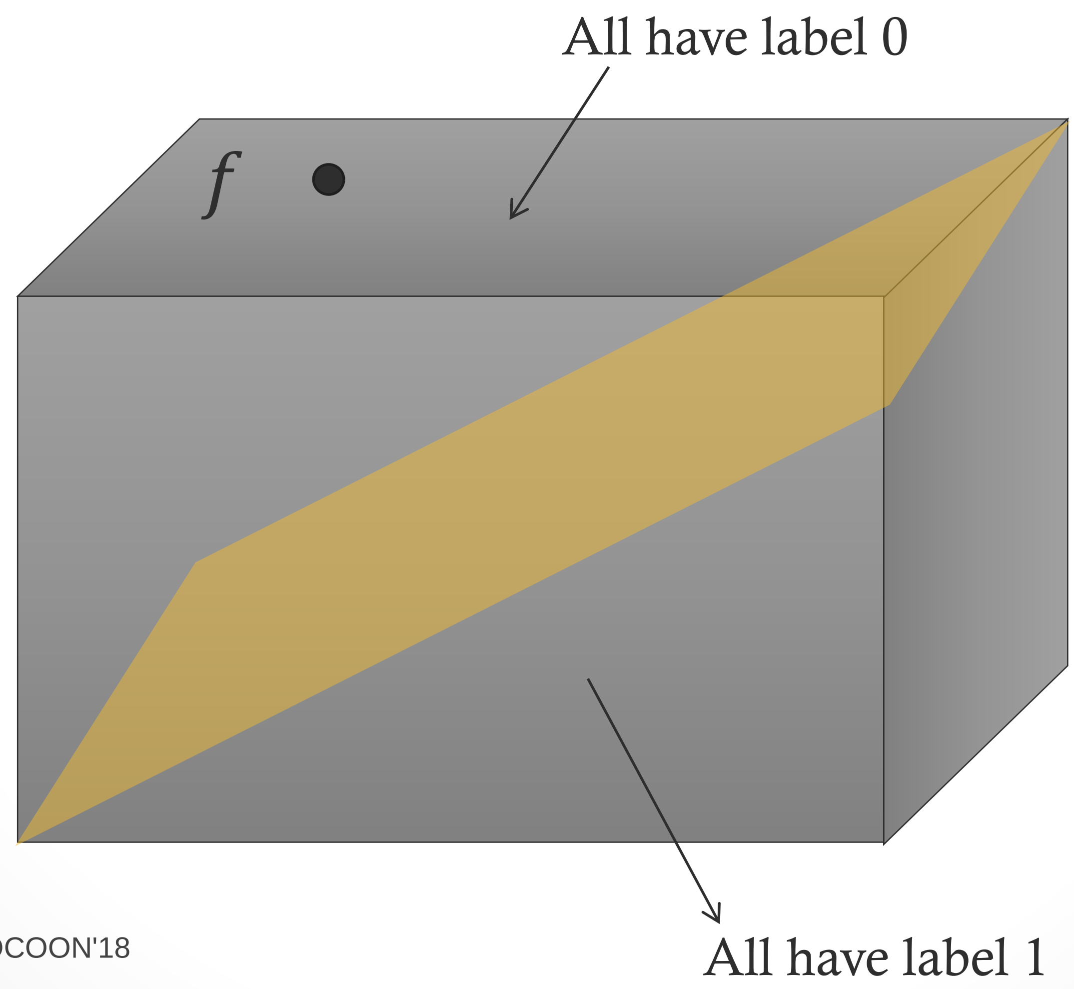
Gaussian Elimination: Geometrically

Consider the vector space $\mathbb{F}_2^n$

An unlabeled example x specifies a hyper plane

The hyperplane divides the space into two halves

Gaussian Elimination



***Gaussian Elimination*:**
Geometrically

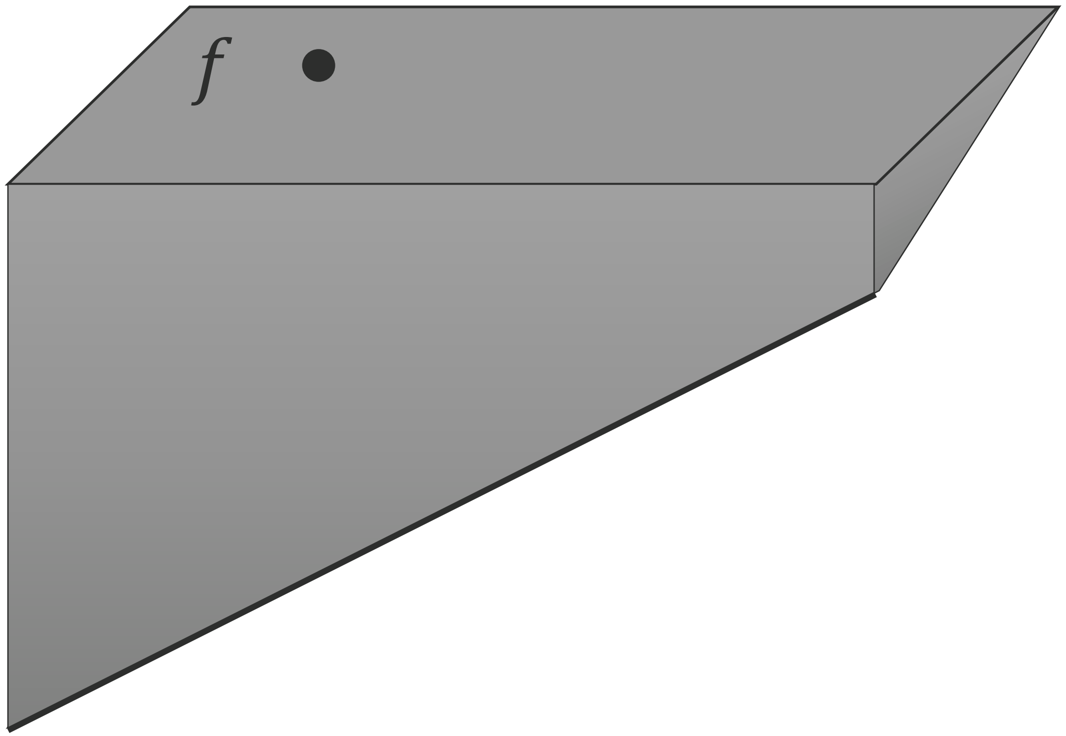
Consider the vector space $\mathbb{F}_2^n$

An unlabeled example x specifies a hyper plane

The hyperplane divides the space into two halves

Predict the majority – say, predicted 1

Gaussian Elimination



Gaussian Elimination: Geometrically

Consider the vector space $\mathbb{F}_2^n$

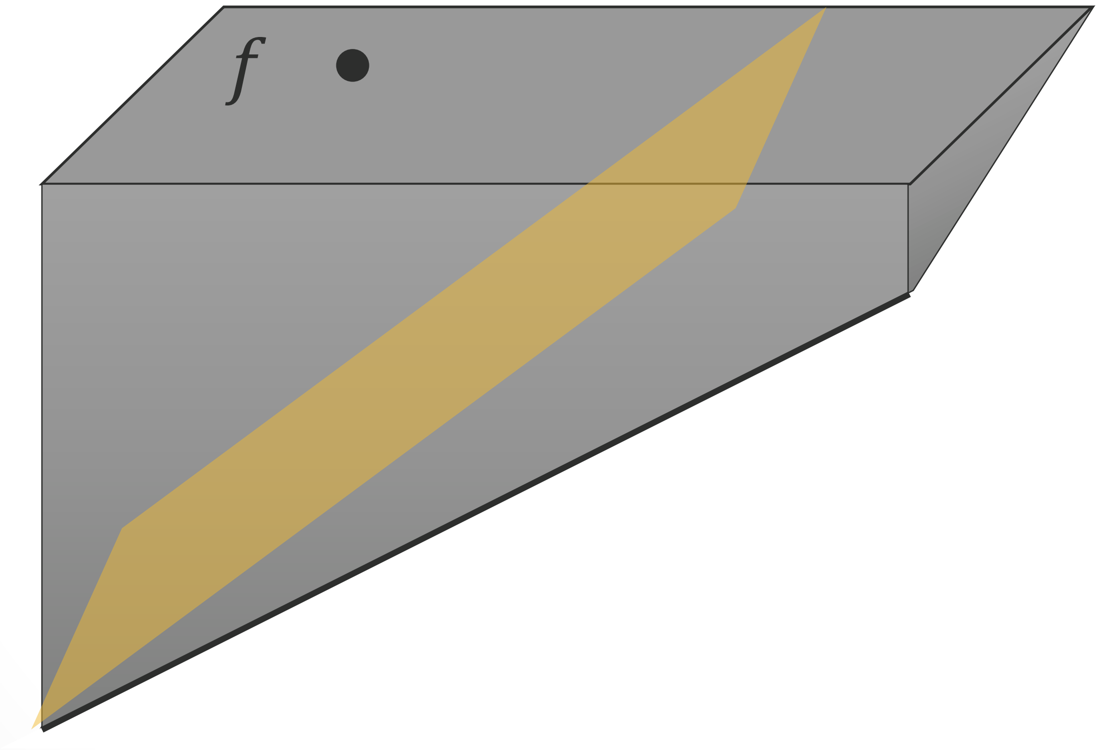
An unlabeled example x specifies a hyper plane

The hyperplane divides the space into two halves

Predict the majority – say, predicted 1

If the true label is 0, throw half space corresponding to 1

Gaussian Elimination



Gaussian Elimination: Geometrically

Consider the vector space $\mathbb{F}_2^n$

An unlabeled example x specifies a hyper plane

The hyperplane divides the space into two halves

Predict the majority – say, predicted 1

If the true label is 0, throw half space corresponding to 1

Repeat with new example

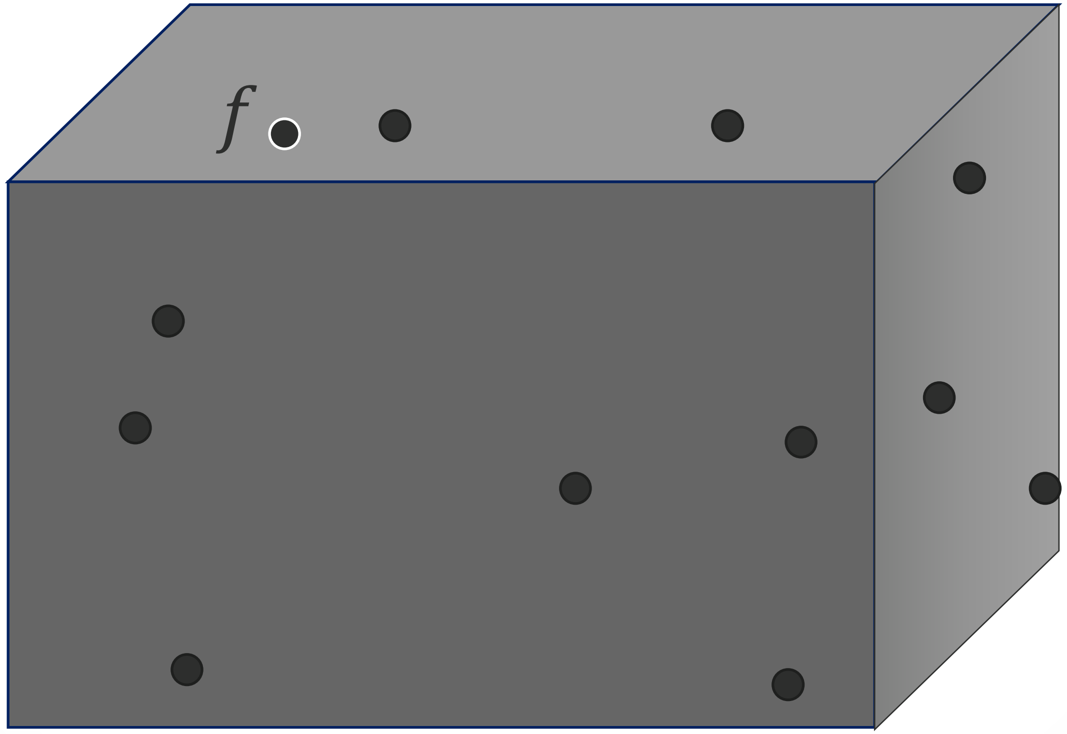
Gaussian Elimination

- *Gaussian Elimination* - Analysis
  - Start with **one set** containing 2^n vectors as possible solutions
  - **Predict** the majority of the labels of the remaining solutions by performing the **intersection of the halfspace** with the remaining subset
  - At each mistake, throw **at least half** of the vectors

Gaussian Elimination

- *Gaussian Elimination* - Analysis
  - After, at most $\log_2 2^n = n$ mistakes, only 1 vector remains = hidden vector f
  - Computing intersection in time $\mathrm{O}(n^3)$ by Gaussian elimination

Halving Algorithm

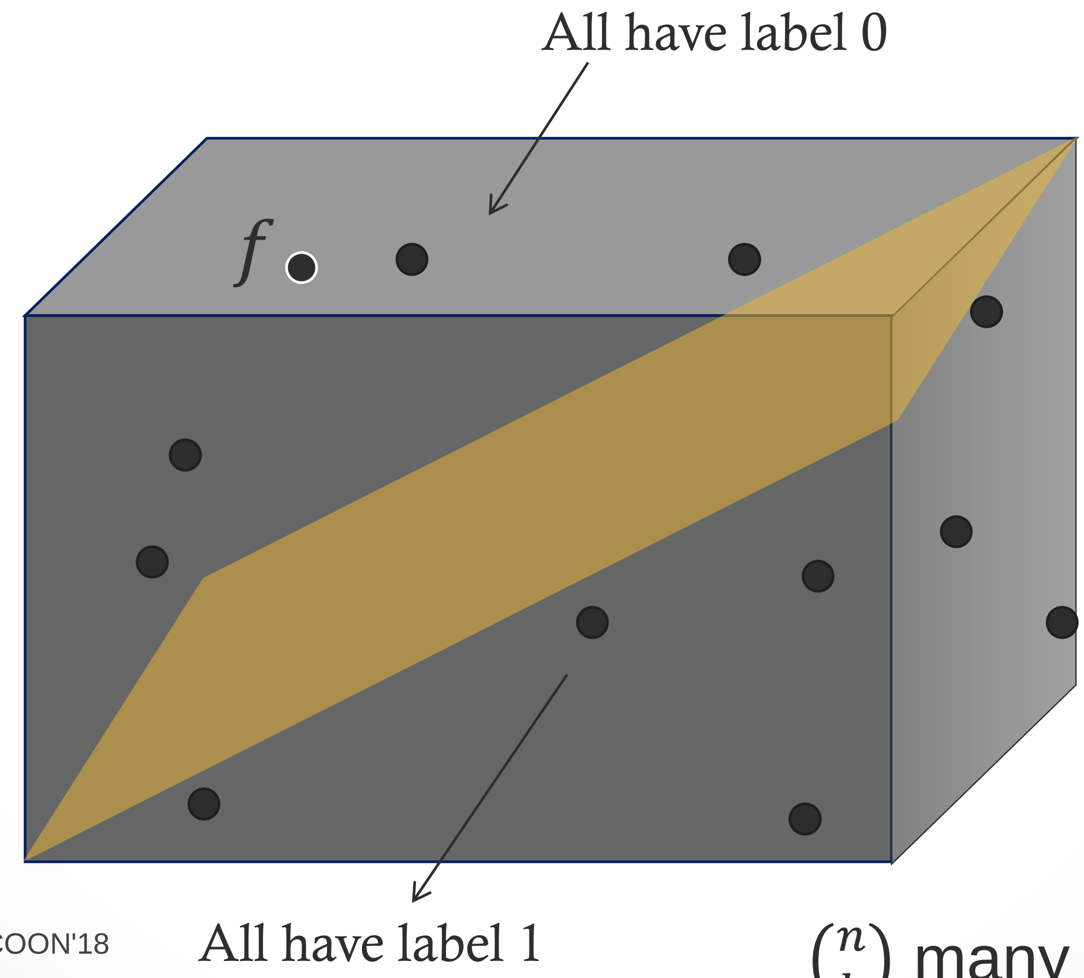


$\binom{n}{k}$ many points

***Halving Algorithm*:**
Geometrically

Consider all k-sparse vectors in vector space $\mathbb{F}_n^2$

Halving Algorithm



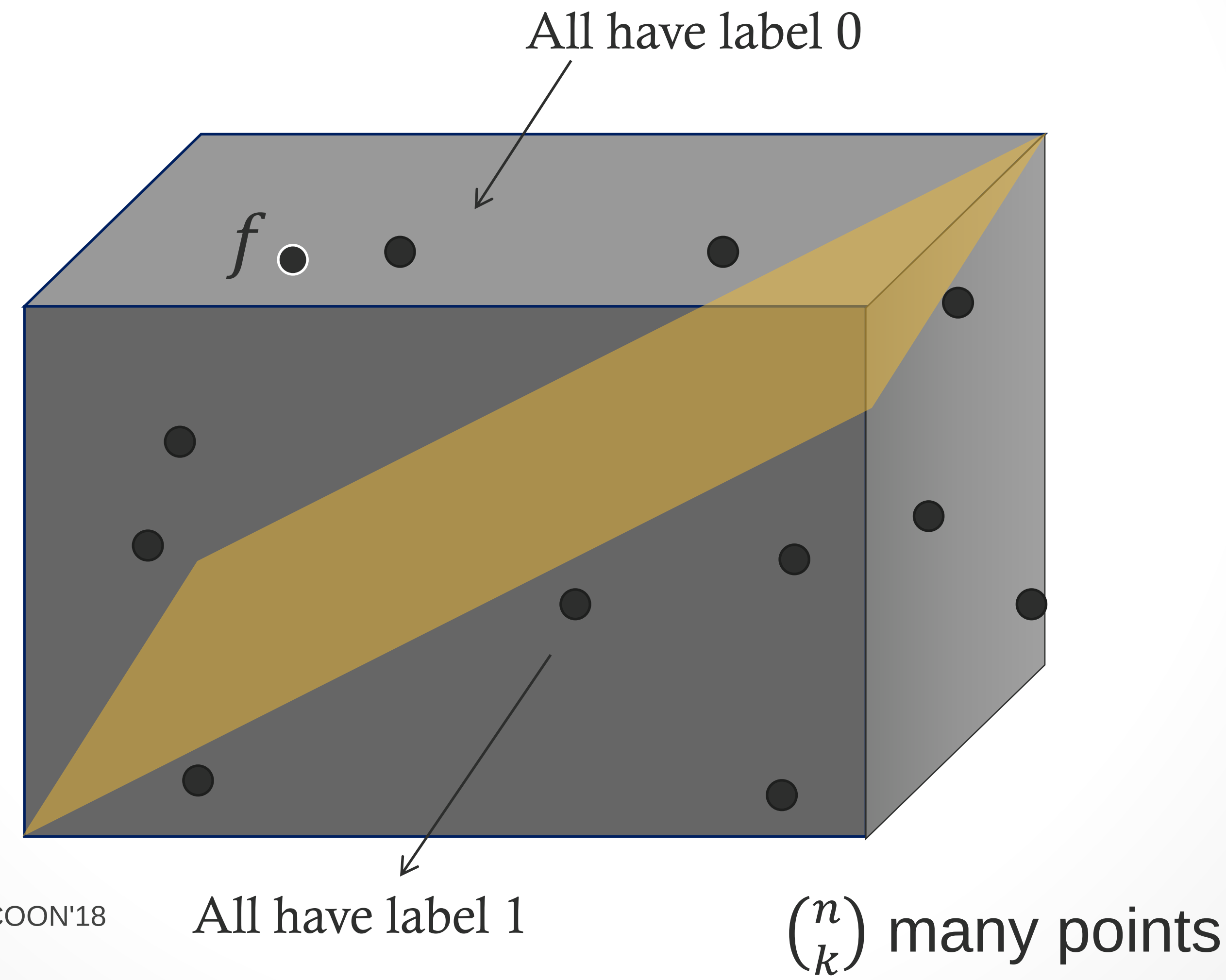
$\binom{n}{k}$ many points

Halving Algorithm: Geometrically

Consider all k-sparse vectors in vector space $\mathbb{F}_n^2$

An unlabeled example x specifies a hyper plane

Halving Algorithm



Halving Algorithm: Geometrically

Consider all k-sparse vectors in vector space $\mathbb{F}_n^2$

An unlabeled example x specifies a hyper plane

Predict the majority – say, predicted 1

If the true label is 0, throw half space corresponding to 1

Repeat with new example

Halving Algorithm

- *Halving Algorithm* - Analysis
  - Start with $\binom{n}{k}$ **sets** as possible solutions such that each k-sparse vector is in one subset.
  - **Predict** the majority of the labels of the remaining solutions by performing the **intersection of the halfspace** with the remaining subset
  - At each mistake, throw **at least half** of the vectors

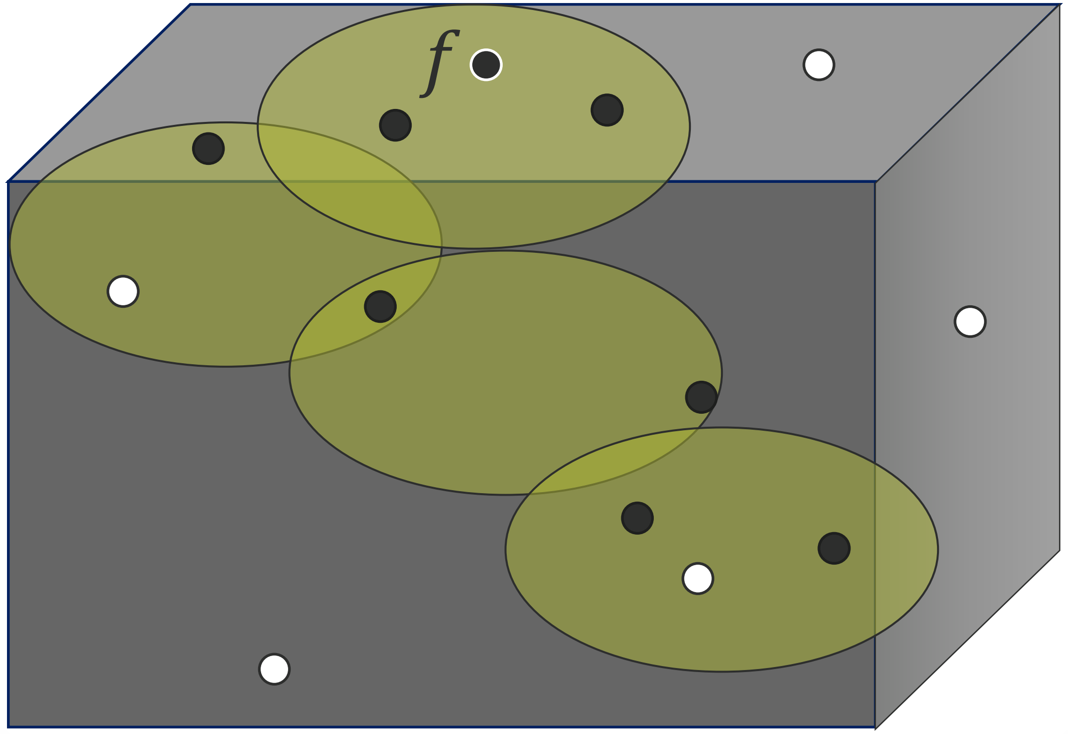
Halving Algorithm

- *Halving Algorithm* - Analysis
  - After, at most $\log_2\binom{n}{k} = k\log n$ mistakes, only 1 vector remains which is the hidden vector f
  - Computing the intersection with all the sets in time $\binom{n}{k}$

The BGM algorithm

- Tries to **balance** both the extremes
- Consider a set of **fewer subsets** such that each k-sparse vector in at least one subset.
- Predict the label which has more **weighted majority** of subsets where weights are proportional to their sizes

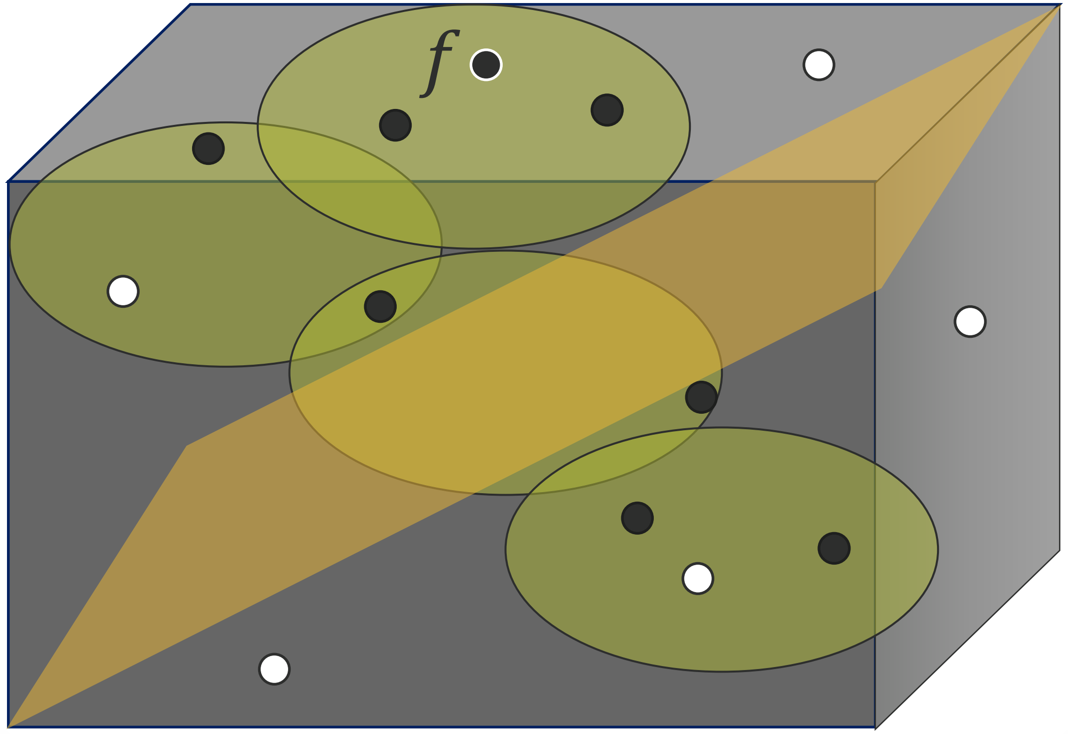
The BGM algorithm



***The BGM algorithm*:**
Geometrically

Consider larger subsets of points such that each k-sparse point is present in some subset

The BGM algorithm

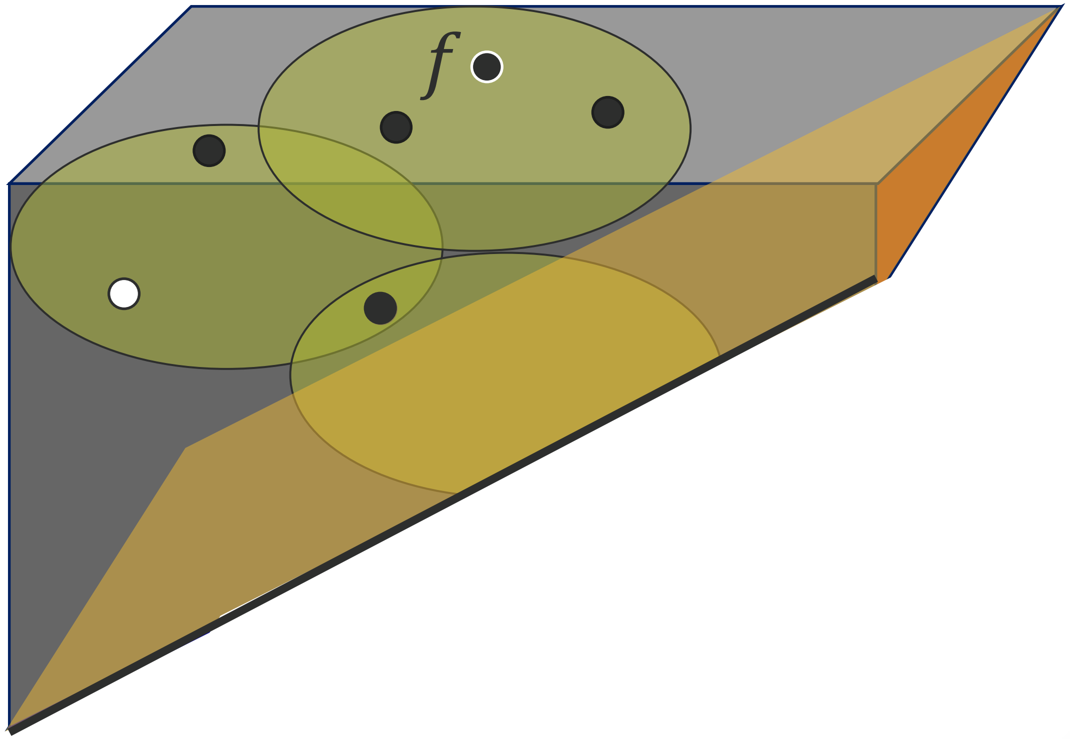


The BGM algorithm: Geometrically

Consider larger subsets of points such that each k-sparse point is present in some subset

An unlabeled example x specifies a hyper plane

The BGM algorithm



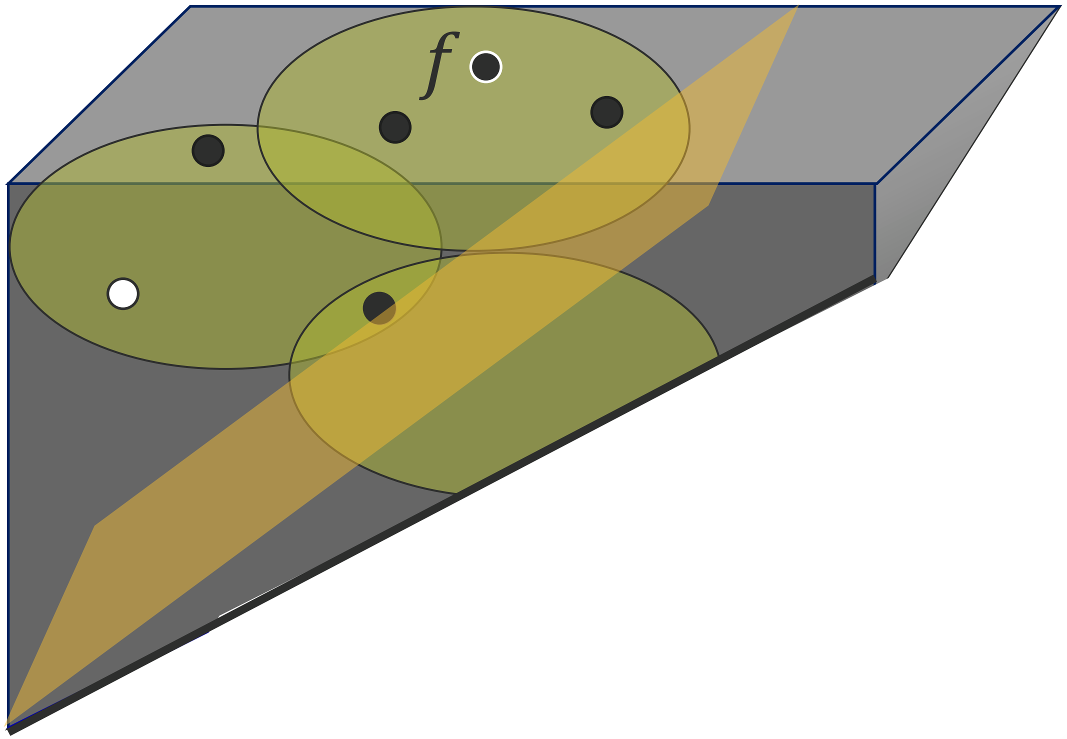
The BGM algorithm: Geometrically

Consider larger subsets of points such that each k-sparse point is present in some subset

An unlabeled example x specifies a hyper plane

Once true label is revealed, throw the irrelevant halfspace

The BGM algorithm



The BGM algorithm: Geometrically

Consider larger subsets of points such that each k-sparse point is present in some subset

An unlabeled example x specifies a hyper plane

Once true label is revealed, throw the irrelevant halfspace

Repeat with next example(or hyper plane)

The BGM algorithm formally

- *Initialization:*
  - Let $f \in \{0,1\}^n$ be the k-sparse parity vector
  - Let $e = \{e_1, e_2, \cdots, e_n\}$ be the set of standard basis vectors of $\{0,1\}^n$
  - Arbitrarily partition e into $t \leq n$ parts $C = C_1, C_2, \cdots, C_t$
  - Let $S \coloneqq k$-subsets of C
  - For each $s \in S$, let M_s be the span of $e_i \in s$. Thus, $|M_s| \leq 2^{k\lceil n/t \rceil}$

The BGM algorithm formally

- *On receiving an example* $x \in \{0,1\}^n$ *:*
  - For each M_s, let $M_s^1 :=$ affine space of $\{M_s\} \cup \{x = 1\}$
  - Similarly, let $M_s^0 :=$ affine space of $\{M_s\} \cup \{x = 0\}$
  - Note that $|M_s^1| = 0, |M_s|$ or $\frac{|M_s|}{2}$
  - Predict $y \in \{0,1\}$ such that $\sum_{s \in S} |M_s^y| \geq \sum_{s \in S} |M_s^{1-y}|$

The BGM algorithm formally

- *On receiving answer* $l \in \{0,1\}$:
  - Update each $M_s = M_s^z$

The BGM algorithm - analysis

- Mistakes:
  - Total number of vectors in the beginning $= \binom{t}{k} 2^{k\lceil n/t \rceil}$
  - At each mistake, throw away at least half of the vectors
  - Number of mistake $\leq \log\left(\binom{t}{k} 2^{k\left\lceil\frac{n}{t}\right\rceil}\right) = \boldsymbol{k\left\lceil\frac{n}{t}\right\rceil + log\binom{t}{k}}$

- Running time:
  - Per Round $\mathbf{O}\left(\binom{\boldsymbol{t}}{\boldsymbol{k}}\left(\frac{\boldsymbol{kn}}{\boldsymbol{t}}\right)^{\mathbf{2}}\right)$

Our Algorithm

- ***Idea*** - Have **slightly bigger** subsets and pick **slightly fewer** of them
  - The setup is same as *BGM, but…..*
  - Partition e into $\boldsymbol{T = 1000t}$ parts $C = C_1, C_2, \cdots, C_T$
  - **Randomly** pick m, $\boldsymbol{1000k}$**-sized** subsets of $[T]$
  - $m = \boldsymbol{O}\left(\frac{\binom{1000t}{1000k}}{\binom{1000t-k}{1000k-k}}\right)$ ensures that with **non zero probability each $\boldsymbol{k}$-sized** subset of $[T]$ is present in some S_i

Our Algorithm

- *Crucial claim:*

$$\frac{\binom{T}{1000k}}{\binom{T-k}{1000k-k}} \leq e^{-k/4.01}\binom{t}{k}$$

- The analysis is same as BGM
- Mistake bound = Mistake bound in BGM *up to constant terms*
- Running time = $e^{-k/4.01}$ × BGM

Relating the results to PAC model

- Standard conversion techniques [*Angluin'88, Littlestone'89, Haussler'88*]
- Allow our result to get an improvement in the PAC model

Learning k-Parity with noise

Learning k-Parity with noise (k-LPN)

- Best known algorithm - *Grigorescu et al. (2011)*

  Time: $\binom{n}{k/2}^{1+4\eta^2+o(1)}$ **Samples:** $\frac{k \log n}{(1-2\eta)^2} \cdot \omega(1)$

- When $\eta \to$ ½, *G. Valiant (2012)* in time $n^{0.8k} \cdot poly\left(\frac{1}{1-2\eta}\right)$

- Barrier of $\binom{n}{k/2}$ in running time!

Breaking the Barrier…

- We show an algorithm that for **polynomially small** but non trivial range of noise rates, it is possible to break this barrier

- For example, when $\eta = \Theta\left(\frac{1}{n^{2/5}}\right)$ and $k = \sqrt{n}$, then our algorithm
  - Runs in time $\boldsymbol{O}\left(\binom{n}{k}^{\frac{1}{4}}\right)$
  - With $\mathbf{O}\left(\boldsymbol{k} \cdot \boldsymbol{n}^{3/8}\right)$ samples

Breaking the Barrier… Algorithm

- Draw **sufficiently** many examples
- Guess a **set** of locations of a particular size (say $\frac{3\eta}{2}$) of the **mis-labelings** and correct them
- Use the **previous learner** from the noiseless setting to get a candidate parity vector
- **Repeat** this for every guess set of that size
- Draw few more examples and pick the **candidate parity vector** which **agrees** with the most number of newly drawn samples

Open Questions

- Noiseless case:
  - poly(n) algorithm with $O\left(\log\binom{n}{k}\right)$ samples – ***attribute efficient learning of parities***
  - Improving our trade-offs
- Noisy case:
  - Lower bounds!
  - Better algorithms [*E.g., Karppa et.al. (2016)*]

Thank you

Attribute efficient learning k-Parity without noise

- Learn k-parity in **polynomial** time with only $\boldsymbol{poly}(\mathbf{log}\binom{n}{k})$ samples

- Best known algorithm using $O\left(\log\binom{n}{k}\right)$ samples, in time $O\left(n^{\frac{k}{2}}\right)$ *[Spielman]*

Our Result for noisy case

Theorem

Suppose $k(n) = n/f(n)$ for some function $f : \mathbb{N} \to \mathbb{N}$ for which $f(n) \ll n/\log\log n$, and suppose $\eta(n) = o(\frac{1}{((f(n))^{\alpha}\log n)})$ for some $\alpha \in [1/2, 1)$. Then, for constant confidence parameter, there exists an algorithm for k-lpn with noise rate η with running time $e^{-k/4.01+o(k)} \cdot \binom{n}{k}^{1-\alpha} \cdot$ poly (n) and sample complexity $O(k(f(n))^{\alpha})$.

For example, consider

$$k = \sqrt{n} \text{ and } \eta = \frac{1}{n^{2/5}} < \frac{1}{n^{3/8}},$$

$$\text{then } \alpha = \frac{3}{4}.$$

In this case, the running time would be $O(\binom{n}{k}^{\frac{1}{4}})$ and the sample complexity would $O(k\binom{n}{k}^{\frac{3}{4}})$.

Learning k-Parity without noise

- Information theoretically, $O\left(\log\binom{n}{k}\right)$ samples
- Running time is $O(n^k)$, improved to $O\left(n^{\frac{k}{2}}\right)$
- **Open question** to get a **polynomial** algorithm with $O\left(\log\binom{n}{k}\right)$ samples

Online Mistake Bound model

- Different than the “black box” model (PAC)
- Learning proceeds in *rounds*
- Each *round*: “**Oracle**” *teaches* the “**Learner**”

Our results for noiseless case

- *Let* $k, t: \mathbb{N} \to \mathbb{N}$ *be two functions such that* $\log\log n \ll k(n) \ll t(n) \ll n$. *Then for every* $n \in \mathbb{N}$, *there is an algorithm that learns k-parity in the mistake-bound model, with mistake bound at most* $\left(1 + o(1)\right)\frac{kn}{t} + \log\binom{t}{k}$ *and running time per round* $e^{-k/4.01} \cdot \binom{t}{k} \cdot \tilde{O}(\left(\frac{kn}{t}\right)^2)$.

BGM'10

- *Let* $k, t: \mathbb{N} \to \mathbb{N}$ *be two functions such that* $k(n) \le t(n) \le n$. *For every* $n \in \mathbb{N}$, *there is a deterministic algorithm that learns k-parity in the mistake-bound model, with mistake bound* $k\left\lceil\frac{n}{t}\right\rceil + \left\lceil\log\binom{t}{k}\right\rceil$ *and running time per round* $O(\binom{t}{k}\left(\frac{kn}{t}\right)^2)$.

Our results for noiseless case

- *Let* $k, t\colon \mathbb{N} \to \mathbb{N}$ *be two functions such that* $\log\log n \ll k(n) \ll t(n) \ll n$. *Then for every* $n \in \mathbb{N}$, *there is an algorithm that learns k-parity in the mistake-bound model, with mistake bound at most* $\left(1 + o(1)\right)\frac{kn}{t} + \log\binom{t}{k}$ *and running time per round* $e^{-k/4.01} \cdot \binom{t}{k} \cdot \tilde{O}\left(\left(\frac{kn}{t}\right)^2\right)$.

BGM'10

- *Let* $k, t\colon \mathbb{N} \to \mathbb{N}$ *be two functions such that* $k(n) \le t(n) \le n$. *For every* $n \in \mathbb{N}$, *there is a deterministic algorithm that learns k-parity in the mistake-bound model, with mistake bound* $k\left\lceil\frac{n}{t}\right\rceil + \left\lceil\log\binom{t}{k}\right\rceil$ *and running time per round* $O\left(\binom{t}{k}\left(\frac{kn}{t}\right)^2\right)$.

Add a Slide Title - 2