Week 2: The combinatorial core of learning

Restriction, growth, shattering, VC dimension, and Sauer–Shelah

Tianhao Wang
tianhaowang@ucsd.edu

UCSD · Spring 2026
DSC 190/291 Topics: Learning Theory

Today

Bridge from Week 1

Restriction and growth

Shattering and VC dimension

Sauer–Shelah lemma

The combinatorial core

What can a hypothesis class do on finitely many points?

Where we left off: online learning

Week 1 was about **online prediction**: an adversary presents examples one by one, and we count mistakes.

Where we left off: online learning

Week 1 was about **online prediction**: an adversary presents examples one by one, and we count mistakes.

Two complexity measures appeared:

Finite classes (Halving)

$$M \leq \log_2 |\mathcal{H}|$$

Complexity = cardinality

Linear predictors (Perceptron)

$$M \leq R^2/\gamma^2$$

Complexity = inverse margin

The gap

The gap

Each measure has a limitation:

- Cardinality: intrinsic to $\mathcal{H}$, but only works for **finite** classes

The gap

Each measure has a limitation:

- Cardinality: intrinsic to $\mathcal{H}$, but only works for **finite** classes
- Margin: works for infinite classes, but **not intrinsic to** $\mathcal{H}$ — it depends on the data

The gap

Each measure has a limitation:

- Cardinality: intrinsic to $\mathcal{H}$, but only works for **finite** classes
- Margin: works for infinite classes, but **not intrinsic to** $\mathcal{H}$ — it depends on the data

Is there a complexity measure that is **intrinsic to** $\mathcal{H}$ and works for **any** class?

Why $|\mathcal{H}|$ is only a proxy

Recall how Halving works: after observing (x_t, y_t), eliminate every h with $h(x_t) \neq y_t$.

Why $|\mathcal{H}|$ is only a proxy

Recall how Halving works: after observing (x_t, y_t), eliminate every h with $h(x_t) \neq y_t$.

Key observation

Two hypotheses that agree on all observed points always survive or die together. So Halving is really tracking **distinct labeling patterns** on the data — not individual hypotheses.

Why $|\mathcal{H}|$ is only a proxy

Recall how Halving works: after observing (x_t, y_t), eliminate every h with $h(x_t) \neq y_t$.

Key observation

Two hypotheses that agree on all observed points always survive or die together. So Halving is really tracking **distinct labeling patterns** on the data — not individual hypotheses.

But how many distinct patterns are there? That depends on which points the adversary picks — we don't know in advance.

Why $|\mathcal{H}|$ is only a proxy

Recall how Halving works: after observing (x_t, y_t), eliminate every h with $h(x_t) \neq y_t$.

Key observation

Two hypotheses that agree on all observed points always survive or die together. So Halving is really tracking **distinct labeling patterns** on the data — not individual hypotheses.

But how many distinct patterns are there? That depends on which points the adversary picks — we don't know in advance.

To make this concrete, let's simplify the setting.

Why $|\mathcal{H}|$ is only a proxy

Recall how Halving works: after observing (x_t, y_t), eliminate every h with $h(x_t) \neq y_t$.

Key observation

Two hypotheses that agree on all observed points always survive or die together. So Halving is really tracking **distinct labeling patterns** on the data — not individual hypotheses.

But how many distinct patterns are there? That depends on which points the adversary picks — we don't know in advance.

To make this concrete, let's simplify the setting.

Suppose there are only n fixed points we will ever need to predict on.

A simpler setting: online transductive learning

Fix an unlabeled pool $C = \{x_1, \ldots, x_n\}$. Assume **realizability**: the labels come from some unknown $h^* \in \mathcal{H}$.

A simpler setting: online transductive learning

Fix an unlabeled pool $C = \{x_1, \ldots, x_n\}$. Assume **realizability**: the labels come from some unknown $h^* \in \mathcal{H}$.

Each round:

- the adversary reveals one point from C,
- the learner predicts its label,
- the true label is revealed.

A simpler setting: online transductive learning

Fix an unlabeled pool $C = \{x_1, \ldots, x_n\}$. Assume **realizability**: the labels come from some unknown $h^* \in \mathcal{H}$.

Each round:

- the adversary reveals one point from C,
- the learner predicts its label,
- the true label is revealed.

In this game, two hypotheses that agree on all points of C are **indistinguishable**.

Restriction of $\mathcal{H}$ to a pool

So the relevant object is the set of distinct labeling patterns:

$$\mathcal{H}|_C = \{(h(x_1), \ldots, h(x_n)) : h \in \mathcal{H}\} \subseteq \{0,1\}^n.$$

Restriction of $\mathcal{H}$ to a pool

So the relevant object is the set of distinct labeling patterns:

$$\mathcal{H}|_C = \{(h(x_1), \ldots, h(x_n)) : h \in \mathcal{H}\} \subseteq \{0,1\}^n.$$

Even if $\mathcal{H}$ is infinite, $\mathcal{H}|_C$ is always finite (at most 2^n patterns).

Restriction of $\mathcal{H}$ to a pool

So the relevant object is the set of distinct labeling patterns:

$$\mathcal{H}|_C = \{(h(x_1), \ldots, h(x_n)) : h \in \mathcal{H}\} \subseteq \{0,1\}^n.$$

Even if $\mathcal{H}$ is infinite, $\mathcal{H}|_C$ is always finite (at most 2^n patterns).

We can run Halving on these patterns.

Algorithm: Halving on restrictions

Maintain a set $S_t \subseteq \mathcal{H}|_C$ of surviving patterns.

Algorithm: Halving on restrictions

Maintain a set $S_t \subseteq \mathcal{H}|_C$ of surviving patterns.

1. Initialize $S_1 = \mathcal{H}|_C$.

Algorithm: Halving on restrictions

Maintain a set $S_t \subseteq \mathcal{H}|_C$ of surviving patterns.

1. Initialize $S_1 = \mathcal{H}|_C$.
2. Adversary reveals a point $x_j \in C$.

Algorithm: Halving on restrictions

Maintain a set $S_t \subseteq \mathcal{H}|_C$ of surviving patterns.

1. Initialize $S_1 = \mathcal{H}|_C$.
2. Adversary reveals a point $x_j \in C$.
3. Predict by majority vote: what do most patterns in S_t say about x_j?

Algorithm: Halving on restrictions

Maintain a set $S_t \subseteq \mathcal{H}|_C$ of surviving patterns.

1. Initialize $S_1 = \mathcal{H}|_C$.
2. Adversary reveals a point $x_j \in C$.
3. Predict by majority vote: what do most patterns in S_t say about x_j?
4. True label y is revealed. Delete every pattern that disagrees:

$$S_{t+1} = \{b \in S_t : b_j = y\}.$$

Algorithm: Halving on restrictions

Maintain a set $S_t \subseteq \mathcal{H}|_C$ of surviving patterns.

1. Initialize $S_1 = \mathcal{H}|_C$.
2. Adversary reveals a point $x_j \in C$.
3. Predict by majority vote: what do most patterns in S_t say about x_j?
4. True label y is revealed. Delete every pattern that disagrees:

$$S_{t+1} = \{b \in S_t : b_j = y\}.$$

Same as Halving from Week 1 — but on **labeling patterns** instead of hypotheses.

Mistake bound for Halving on restrictions

Let $N =$ number of patterns in $\mathcal{H}|_C$.

Mistake bound for Halving on restrictions

Let $N =$ number of patterns in $\mathcal{H}|_C$.

On a **mistake**: the majority was wrong, so at least half the surviving patterns were wrong and get deleted.

Mistake bound for Halving on restrictions

Let $N =$ number of patterns in $\mathcal{H}|_C$.

On a **mistake**: the majority was wrong, so at least half the surviving patterns were wrong and get deleted.

Every mistake cuts the count by at least half:

$$N \to N/2 \to N/4 \to \cdots$$

Mistake bound for Halving on restrictions

Let $N =$ number of patterns in $\mathcal{H}|_C$.

On a **mistake**: the majority was wrong, so at least half the surviving patterns were wrong and get deleted.

Every mistake cuts the count by at least half:

$$N \to N/2 \to N/4 \to \cdots$$

The true pattern is never deleted, so we never go below 1.

Mistake bound for Halving on restrictions

Let $N =$ number of patterns in $\mathcal{H}|_C$.

On a **mistake**: the majority was wrong, so at least half the surviving patterns were wrong and get deleted.

Every mistake cuts the count by at least half:

$$N \to N/2 \to N/4 \to \cdots$$

The true pattern is never deleted, so we never go below 1.

Therefore

$$M \leq \log_2 N.$$

Mistake bound for Halving on restrictions

Let $N =$ number of patterns in $\mathcal{H}|_C$.

On a **mistake**: the majority was wrong, so at least half the surviving patterns were wrong and get deleted.

Every mistake cuts the count by at least half:

$$N \to N/2 \to N/4 \to \cdots$$

The true pattern is never deleted, so we never go below 1.

Therefore

$$M \leq \log_2 N.$$

Compare with the original Halving bound $M \leq \log_2 |\mathcal{H}|$. Now N replaces $|\mathcal{H}|$ — and N is finite even when $|\mathcal{H}| = \infty$.

A first fixed-pool example: thresholds

Consider thresholds $\mathcal{H} = \{x \mapsto \mathbf{1}[x \le \theta] : \theta \in \mathbb{R}\}$. The class is infinite.

A first fixed-pool example: thresholds

Consider thresholds $\mathcal{H} = \{x \mapsto \mathbf{1}[x \le \theta] : \theta \in \mathbb{R}\}$. The class is infinite.
Suppose the pool contains three ordered points $x_1 < x_2 < x_3$. What patterns can $\mathcal{H}$ produce?

A first fixed-pool example: thresholds

Consider thresholds $\mathcal{H} = \{x \mapsto \mathbf{1}[x \le \theta] : \theta \in \mathbb{R}\}$. The class is infinite.
Suppose the pool contains three ordered points $x_1 < x_2 < x_3$. What patterns can $\mathcal{H}$ produce?

θ range	$h(x_1)$	$h(x_2)$	$h(x_3)$
$\theta < x_1$	0	0	0
$x_1 \le \theta < x_2$	1	0	0
$x_2 \le \theta < x_3$	1	1	0
$\theta \ge x_3$	1	1	1

A first fixed-pool example: thresholds

Consider thresholds $\mathcal{H} = \{x \mapsto \mathbf{1}[x \le \theta] : \theta \in \mathbb{R}\}$. The class is infinite.
Suppose the pool contains three ordered points $x_1 < x_2 < x_3$. What patterns can $\mathcal{H}$ produce?

θ range	$h(x_1)$	$h(x_2)$	$h(x_3)$
$\theta < x_1$	0	0	0
$x_1 \le \theta < x_2$	1	0	0
$x_2 \le \theta < x_3$	1	1	0
$\theta \ge x_3$	1	1	1

So the only possible labelings on this pool are

$$000, \quad 100, \quad 110, \quad 111.$$

Running the algorithm: thresholds

True labeling: 000 ($\theta^* < x_1$). Reveal order: x_2, x_1, x_3.

Running the algorithm: thresholds

True labeling: 000 ($\theta^* < x_1$). Reveal order: x_2, x_1, x_3.

round	point	majority prediction	surviving patterns
			$\{000, 100, 110, 111\}$

Running the algorithm: thresholds

True labeling: 000 ($\theta^* < x_1$). Reveal order: x_2, x_1, x_3.

round	point	majority prediction	surviving patterns
			$\{000, 100, 110, 111\}$
1	x_2	tie, predict 1	$\times$ $\{000, 100\}$

Running the algorithm: thresholds

True labeling: 000 ($\theta^* < x_1$). Reveal order: x_2, x_1, x_3.

round	point	majority prediction	surviving patterns
			$\{000, 100, 110, 111\}$
1	x_2	tie, predict 1 $\times$	$\{000, 100\}$
2	x_1	tie, predict 1 $\times$	$\{000\}$

Running the algorithm: thresholds

True labeling: 000 ($\theta^* < x_1$). Reveal order: x_2, x_1, x_3.

round	point	majority prediction	surviving patterns
			$\{000, 100, 110, 111\}$
1	x_2	tie, predict 1 $\times$	$\{000, 100\}$
2	x_1	tie, predict 1 $\times$	$\{000\}$
3	x_3	0 ✓	$\{000\}$

Running the algorithm: thresholds

True labeling: 000 ($\theta^* < x_1$). Reveal order: x_2, x_1, x_3.

round	point	majority prediction	surviving patterns
			$\{000, 100, 110, 111\}$
1	x_2	tie, predict 1 $\times$	$\{000, 100\}$
2	x_1	tie, predict 1 $\times$	$\{000\}$
3	x_3	0 ✓	$\{000\}$

Each mistake halves the surviving set: $4 \to 2 \to 1$.

Two mistakes — exactly $\log_2 4 = 2$. The bound is tight here.

The right question

The mistake bound on pool C depends on the **number of patterns** — not on $|\mathcal{H}|$.

The right question

The mistake bound on pool C depends on the **number of patterns** — not on $|\mathcal{H}|$.

But this number depends on the specific pool — and the **adversary** gets to choose it.

The right question

The mistake bound on pool C depends on the **number of patterns** — not on $|\mathcal{H}|$.

But this number depends on the specific pool — and the **adversary** gets to choose it.

So the learner must be prepared for the **worst case**:

$$\max_{C \in \mathcal{X}^n} \text{ number of distinct patterns in } \mathcal{H}|_C.$$

The right question

The mistake bound on pool C depends on the **number of patterns** — not on $|\mathcal{H}|$.

But this number depends on the specific pool — and the **adversary** gets to choose it.

So the learner must be prepared for the **worst case**:

$$\max_{C \in \mathcal{X}^n} \text{ number of distinct patterns in } \mathcal{H}|_C.$$

How many **distinct behaviors** can $\mathcal{H}$ produce on n points in the worst case?

Three questions for this week

The mistake bound is $M \leq \log_2 N$, where $N =$ number of patterns. This leads to three questions:

1. **What is N?** How many distinct patterns can $\mathcal{H}$ produce on n points?

Three questions for this week

The mistake bound is $M \leq \log_2 N$, where $N =$ number of patterns. This leads to three questions:

1. **What is N?** How many distinct patterns can $\mathcal{H}$ produce on n points?
2. **When is $N = 2^n$?** Then the bound is n — no better than no assumption. For which n does this happen?

Three questions for this week

The mistake bound is $M \leq \log_2 N$, where $N =$ number of patterns. This leads to three questions:

1. **What is N?** How many distinct patterns can $\mathcal{H}$ produce on n points?
2. **When is $N = 2^n$?** Then the bound is n — no better than no assumption. For which n does this happen?
3. **How fast does N drop below 2^n?** Once the learner has an advantage, is it marginal or dramatic?

Three questions for this week

The mistake bound is $M \leq \log_2 N$, where $N =$ number of patterns. This leads to three questions:

1. **What is N?** How many distinct patterns can $\mathcal{H}$ produce on n points?
2. **When is $N = 2^n$?** Then the bound is n — no better than no assumption. For which n does this happen?
3. **How fast does N drop below 2^n?** Once the learner has an advantage, is it marginal or dramatic?

restriction $\rightarrow$ growth function $\rightarrow$ shattering $\rightarrow$ VC dimension $\rightarrow$ Sauer–Shelah.

Restriction and growth

Count what the class can do on a fixed finite pool

Restriction to a finite set

We saw that thresholds produce only 4 patterns on 3 points. Let's give this idea a name.

Restriction to a finite set

We saw that thresholds produce only 4 patterns on 3 points. Let's give this idea a name.

Definition (restriction)

Let $C = (x_1, \dots, x_n) \in \mathcal{X}^n$. The **restriction** of $\mathcal{H}$ to C is

$$\mathcal{H}|_C = \{(h(x_1), \dots, h(x_n)) \in \{0,1\}^n : h \in \mathcal{H}\}.$$

Restriction to a finite set

We saw that thresholds produce only 4 patterns on 3 points. Let's give this idea a name.

Definition (restriction)

Let $C = (x_1, \ldots, x_n) \in \mathcal{X}^n$. The **restriction** of $\mathcal{H}$ to C is

$$\mathcal{H}|_C = \{(h(x_1), \ldots, h(x_n)) \in \{0,1\}^n : h \in \mathcal{H}\}.$$

For thresholds on $x_1 < x_2 < x_3$:

$$\mathcal{H}|_C = \{000,\ 100,\ 110,\ 111\} \qquad \text{— four patterns.}$$

The growth function

Definition

For a finite set C, let $\Gamma_{\mathcal{H}}(C)$ be the number of distinct patterns in $\mathcal{H}|_C$.

The **growth function** is

$$\Gamma_{\mathcal{H}}(n) = \max_{C \in \mathcal{X}^n} \Gamma_{\mathcal{H}}(C).$$

The growth function

Definition

For a finite set C, let $\Gamma_{\mathcal{H}}(C)$ be the number of distinct patterns in $\mathcal{H}|_C$.

The **growth function** is

$$\Gamma_{\mathcal{H}}(n) = \max_{C \in \mathcal{X}^n} \Gamma_{\mathcal{H}}(C).$$

Note: $\Gamma_{\mathcal{H}}(C)$ is about a specific set; $\Gamma_{\mathcal{H}}(n)$ maximizes over all sets of size n.

The growth function

Definition

For a finite set C, let $\Gamma_{\mathcal{H}}(C)$ be the number of distinct patterns in $\mathcal{H}|_C$.

The **growth function** is

$$\Gamma_{\mathcal{H}}(n) = \max_{C \in \mathcal{X}^n} \Gamma_{\mathcal{H}}(C).$$

Note: $\Gamma_{\mathcal{H}}(C)$ is about a specific set; $\Gamma_{\mathcal{H}}(n)$ maximizes over all sets of size n.

This is exactly the adversary's worst case from earlier: the n points where $\mathcal{H}$ has the most distinct behaviors.

The growth function

Definition

For a finite set C, let $\Gamma_{\mathcal{H}}(C)$ be the number of distinct patterns in $\mathcal{H}|_C$.

The **growth function** is

$$\Gamma_{\mathcal{H}}(n) = \max_{C \in \mathcal{X}^n} \Gamma_{\mathcal{H}}(C).$$

Note: $\Gamma_{\mathcal{H}}(C)$ is about a specific set; $\Gamma_{\mathcal{H}}(n)$ maximizes over all sets of size n.

This is exactly the adversary's worst case from earlier: the n points where $\mathcal{H}$ has the most distinct behaviors.

Always $1 \leq \Gamma_{\mathcal{H}}(n) \leq 2^n$.

Example 1: the unrestricted class

Let $\mathcal{H} = \{h : \mathcal{X} \to \{0, 1\}\}$, the class of *all* binary functions on the domain.

Example 1: the unrestricted class

Let $\mathcal{H} = \{h : \mathcal{X} \to \{0, 1\}\}$, the class of *all* binary functions on the domain.

On any n distinct points, every labeling is achievable:

$$\Gamma_{\mathcal{H}}(n) = 2^n.$$

Example 1: the unrestricted class

Let $\mathcal{H} = \{h : \mathcal{X} \to \{0, 1\}\}$, the class of *all* binary functions on the domain.

On any n distinct points, every labeling is achievable:

$$\Gamma_{\mathcal{H}}(n) = 2^n.$$

Why? For any target pattern on $x_1, \ldots, x_n$, just define a function that matches it.

Example 1: the unrestricted class

Let $\mathcal{H} = \{h : \mathcal{X} \to \{0, 1\}\}$, the class of *all* binary functions on the domain.

On any n distinct points, every labeling is achievable:

$$\Gamma_{\mathcal{H}}(n) = 2^n.$$

Why? For any target pattern on $x_1, \ldots, x_n$, just define a function that matches it.

This is the maximum possible growth — the class has no structure at all.

Example 2: thresholds on the line

We saw this on 3 points previously. In general, for

$$\mathcal{H} = \{x \mapsto \mathbf{1}[x \leq \theta] : \theta \in \mathbb{R}\},$$

Example 2: thresholds on the line

We saw this on 3 points previously. In general, for

$$\mathcal{H} = \{x \mapsto \mathbf{1}[x \le \theta] : \theta \in \mathbb{R}\},$$

if $x_1 < x_2 < \cdots < x_n$, the only patterns are

$$000\cdots0,\ 100\cdots0,\ 110\cdots0,\ \ldots,\ 111\cdots1.$$

Example 2: thresholds on the line

We saw this on 3 points previously. In general, for

$$\mathcal{H} = \{x \mapsto \mathbf{1}[x \le \theta] : \theta \in \mathbb{R}\},$$

if $x_1 < x_2 < \cdots < x_n$, the only patterns are

$$000\cdots0,\ 100\cdots0,\ 110\cdots0,\ \ldots,\ 111\cdots1.$$

Therefore

$$\Gamma_{\mathcal{H}}(n) = n + 1.$$

Example 2: thresholds on the line

We saw this on 3 points previously. In general, for

$$\mathcal{H} = \{x \mapsto \mathbf{1}[x \le \theta] : \theta \in \mathbb{R}\},$$

if $x_1 < x_2 < \cdots < x_n$, the only patterns are

$$000\cdots0,\ 100\cdots0,\ 110\cdots0,\ \ldots,\ 111\cdots1.$$

Therefore

$$\Gamma_{\mathcal{H}}(n) = n + 1.$$

Compare: the unrestricted class has $\Gamma_{\mathcal{H}}(n) = 2^n$. Thresholds have $\Gamma_{\mathcal{H}}(n) = n + 1$. Growth captures this difference — cardinality does not ($|\mathcal{H}| = \infty$ for both).

Example 3: intervals on the line

Consider

$$\mathcal{H} = \{x \mapsto \mathbf{1}[a \le x \le b] : a, b \in \mathbb{R}\}.$$

Example 3: intervals on the line

Consider

$$\mathcal{H} = \{x \mapsto \mathbf{1}[a \le x \le b] : a, b \in \mathbb{R}\}.$$

If $x_1 < \cdots < x_n$, any realized positive set must be a **single contiguous block**:

$$00\cdots0\underbrace{11\cdots1}_{\text{block}}00\cdots0.$$

Example 3: intervals on the line

Consider

$$\mathcal{H} = \{x \mapsto \mathbf{1}[a \le x \le b] : a, b \in \mathbb{R}\}.$$

If $x_1 < \cdots < x_n$, any realized positive set must be a **single contiguous block**:

$$00\cdots0\underbrace{11\cdots1}_{\text{block}}00\cdots0.$$

Counting: choose where the block starts and ends among the n positions, plus the all-zeros pattern:

$$\Gamma_{\mathcal{H}}(n) = 1 + \binom{n+1}{2} \sim \frac{n^2}{2}.$$

Example 3: intervals on the line

Consider

$$\mathcal{H} = \{x \mapsto \mathbf{1}[a \le x \le b] : a, b \in \mathbb{R}\}.$$

If $x_1 < \cdots < x_n$, any realized positive set must be a **single contiguous block**:

$$00\cdots0\underbrace{11\cdots1}_{\text{block}}00\cdots0.$$

Counting: choose where the block starts and ends among the n positions, plus the all-zeros pattern:

$$\Gamma_{\mathcal{H}}(n) = 1 + \binom{n+1}{2} \sim \frac{n^2}{2}.$$

Another infinite class, but growth is polynomial, not exponential.

Basic properties of the growth function

Recall: $\Gamma_{\mathcal{H}}(n)$ is the most distinct patterns $\mathcal{H}$ can produce on any n points.

- **Monotone in n:** allowing more sample points cannot decrease maximal behavior.

Basic properties of the growth function

Recall: $\Gamma_{\mathcal{H}}(n)$ is the most distinct patterns $\mathcal{H}$ can produce on any n points.

- **Monotone in n:** allowing more sample points cannot decrease maximal behavior.
- **Cardinality bound:**

$$\Gamma_{\mathcal{H}}(n) \leq |\mathcal{H}|.$$

  This may be very loose.

Basic properties of the growth function

Recall: $\Gamma_{\mathcal{H}}(n)$ is the most distinct patterns $\mathcal{H}$ can produce on any n points.

- **Monotone in n:** allowing more sample points cannot decrease maximal behavior.
- **Cardinality bound:**

$$\Gamma_{\mathcal{H}}(n) \leq |\mathcal{H}|.$$

  This may be very loose.
- **Universal upper bound:**

$$\Gamma_{\mathcal{H}}(n) \leq 2^n.$$

Basic properties of the growth function

Recall: $\Gamma_{\mathcal{H}}(n)$ is the most distinct patterns $\mathcal{H}$ can produce on any n points.

- **Monotone in n:** allowing more sample points cannot decrease maximal behavior.
- **Cardinality bound:**
$$\Gamma_{\mathcal{H}}(n) \leq |\mathcal{H}|.$$
This may be very loose.
- **Universal upper bound:**
$$\Gamma_{\mathcal{H}}(n) \leq 2^n.$$

- **Infinite classes can still be simple:** thresholds already showed this.

Basic properties of the growth function

Recall: $\Gamma_{\mathcal{H}}(n)$ is the most distinct patterns $\mathcal{H}$ can produce on any n points.

- **Monotone in n:** allowing more sample points cannot decrease maximal behavior.
- **Cardinality bound:**
$$\Gamma_{\mathcal{H}}(n) \leq |\mathcal{H}|.$$
This may be very loose.
- **Universal upper bound:**
$$\Gamma_{\mathcal{H}}(n) \leq 2^n.$$

- **Infinite classes can still be simple:** thresholds already showed this.

We know $\Gamma_{\mathcal{H}}(n) \leq 2^n$. When does equality hold — when does $\mathcal{H}$ realize *all* 2^n labelings?

Shattering and VC dimension

When does the class behave like all possible labelings?

Shattering

Recall question 2: when is $N = 2^n$?

Shattering

Recall question 2: when is $N = 2^n$?

That's the worst case for the learner — the mistake bound is $\log_2 2^n = n$, no better than having no assumption at all.

Shattering

Recall question 2: when is $N = 2^n$?

That's the worst case for the learner — the mistake bound is $\log_2 2^n = n$, no better than having no assumption at all.

Definition

A pool $C = \{x_1, \ldots, x_n\}$ is **shattered** by $\mathcal{H}$ if $\mathcal{H}$ realizes every labeling of C:

$$\text{for every } (y_1, \ldots, y_n) \in \{0,1\}^n, \text{ there exists } h \in \mathcal{H} \text{ with } h(x_i) = y_i \; \forall\, i.$$

Shattering

Recall question 2: when is $N = 2^n$?

That's the worst case for the learner — the mistake bound is $\log_2 2^n = n$, no better than having no assumption at all.

Definition

A pool $C = \{x_1, \ldots, x_n\}$ is **shattered** by $\mathcal{H}$ if $\mathcal{H}$ realizes every labeling of C:

$$\text{for every } (y_1, \ldots, y_n) \in \{0,1\}^n, \text{ there exists } h \in \mathcal{H} \text{ with } h(x_i) = y_i \; \forall\, i.$$

On a shattered pool, the class has **complete freedom** — the learner gains nothing from knowing $\mathcal{H}$.

What shattering means for the learner

If $\mathcal{H}$ **shatters** a set C of size n:

- Every labeling is consistent with some $h \in \mathcal{H}$.

What shattering means for the learner

If $\mathcal{H}$ **shatters** a set C of size n:

- Every labeling is consistent with some $h \in \mathcal{H}$.
- The adversary can pick C, and the learner faces all 2^n possibilities.

What shattering means for the learner

If $\mathcal{H}$ **shatters** a set C of size n:

- Every labeling is consistent with some $h \in \mathcal{H}$.
- The adversary can pick C, and the learner faces all 2^n possibilities.
- No structural advantage — as hard as having no assumption at all.

What shattering means for the learner

If $\mathcal{H}$ **shatters** a set C of size n:

- Every labeling is consistent with some $h \in \mathcal{H}$.
- The adversary can pick C, and the learner faces all 2^n possibilities.
- No structural advantage — as hard as having no assumption at all.

If $\mathcal{H}$ **cannot shatter** any set of size n:

- On *every* set of n points, some labelings are impossible.

What shattering means for the learner

If $\mathcal{H}$ **shatters** a set C of size n:

- Every labeling is consistent with some $h \in \mathcal{H}$.
- The adversary can pick C, and the learner faces all 2^n possibilities.
- No structural advantage — as hard as having no assumption at all.

If $\mathcal{H}$ **cannot shatter** any set of size n:

- On *every* set of n points, some labelings are impossible.
- The learner always has constraints to exploit.

The breakpoint

Can thresholds $\mathcal{H} = \{x \mapsto \mathbf{1}[x \leq \theta] : \theta \in \mathbb{R}\}$ shatter 1 point?

The breakpoint

Can thresholds $\mathcal{H} = \{x \mapsto \mathbf{1}[x \leq \theta] : \theta \in \mathbb{R}\}$ shatter 1 point?

Yes — any single point can be labeled 0 or 1 by choosing θ.

The breakpoint

Can thresholds $\mathcal{H} = \{x \mapsto \mathbf{1}[x \leq \theta] : \theta \in \mathbb{R}\}$ shatter 1 point?

Yes — any single point can be labeled 0 or 1 by choosing θ.

Can they shatter 2 points $x_1 < x_2$?

The breakpoint

Can thresholds $\mathcal{H} = \{x \mapsto \mathbf{1}[x \leq \theta] : \theta \in \mathbb{R}\}$ shatter 1 point?

Yes — any single point can be labeled 0 or 1 by choosing θ.

Can they shatter 2 points $x_1 < x_2$?

No — the patterns are $00, 10, 11$. The labeling $0, 1$ is impossible.

The breakpoint

Can thresholds $\mathcal{H} = \{x \mapsto \mathbf{1}[x \le \theta] : \theta \in \mathbb{R}\}$ shatter 1 point?

Yes — any single point can be labeled 0 or 1 by choosing θ.

Can they shatter 2 points $x_1 < x_2$?

No — the patterns are $00, 10, 11$. The labeling $0, 1$ is impossible.

So there is a **breakpoint** at 1:

- Below it, $\mathcal{H}$ can still mimic the unrestricted class on the right points.
- Above it, $\mathcal{H}$ is always constrained.

The breakpoint

Can thresholds $\mathcal{H} = \{x \mapsto \mathbf{1}[x \leq \theta] : \theta \in \mathbb{R}\}$ shatter 1 point?

Yes — any single point can be labeled 0 or 1 by choosing θ.

Can they shatter 2 points $x_1 < x_2$?

No — the patterns are $00, 10, 11$. The labeling $0, 1$ is impossible.

So there is a **breakpoint** at 1:

- Below it, $\mathcal{H}$ can still mimic the unrestricted class on the right points.
- Above it, $\mathcal{H}$ is always constrained.

This breakpoint is intrinsic to $\mathcal{H}$ and works for any class — exactly the complexity measure we wanted.

VC dimension

Definition

The **VC dimension** of $\mathcal{H}$, written $\mathrm{VCdim}(\mathcal{H})$, is the largest n such that some $C \subseteq \mathcal{X}$ of size n is shattered by $\mathcal{H}$.

If arbitrarily large sets can be shattered, write $\mathrm{VCdim}(\mathcal{H}) = \infty$.

Named after Vapnik and Chervonenkis (1971).

Definition

The **VC dimension** of $\mathcal{H}$, written $\mathrm{VCdim}(\mathcal{H})$, is the largest n such that some $C \subseteq \mathcal{X}$ of size n is shattered by $\mathcal{H}$.

If arbitrarily large sets can be shattered, write $\mathrm{VCdim}(\mathcal{H}) = \infty$.

Named after Vapnik and Chervonenkis (1971).

For thresholds: can shatter 1 point, cannot shatter 2. So $\mathrm{VCdim}(\mathcal{H}) = 1$.

Definition

The **VC dimension** of $\mathcal{H}$, written $\mathrm{VCdim}(\mathcal{H})$, is the largest n such that some $C \subseteq \mathcal{X}$ of size n is shattered by $\mathcal{H}$.

If arbitrarily large sets can be shattered, write $\mathrm{VCdim}(\mathcal{H}) = \infty$.

Named after Vapnik and Chervonenkis (1971).

For thresholds: can shatter 1 point, cannot shatter 2. So $\mathrm{VCdim}(\mathcal{H}) = 1$.

For the unrestricted class: can shatter any finite set. So $\mathrm{VCdim}(\mathcal{H}) = \infty$.

VC example: intervals

Consider intervals $\mathcal{H} = \{x \mapsto \mathbf{1}[a \le x \le b] : a, b \in \mathbb{R}\}$.

VC example: intervals

Consider intervals $\mathcal{H} = \{x \mapsto \mathbf{1}[a \le x \le b] : a, b \in \mathbb{R}\}$.

Lower bound ($\mathrm{VCdim} \ge 2$): pick two points $x_1 < x_2$. All four labelings are achievable:

$$00, \quad 10, \quad 01, \quad 11.$$

VC example: intervals

Consider intervals $\mathcal{H} = \{x \mapsto \mathbf{1}[a \le x \le b] : a, b \in \mathbb{R}\}$.

Lower bound ($\mathrm{VCdim} \ge 2$): pick two points $x_1 < x_2$. All four labelings are achievable:

$$00, \quad 10, \quad 01, \quad 11.$$

Upper bound ($\mathrm{VCdim} \le 2$): for any three ordered points $x_1 < x_2 < x_3$, the labeling $1, 0, 1$ is impossible — an interval can't skip the middle point.

VC example: intervals

Consider intervals $\mathcal{H} = \{x \mapsto \mathbf{1}[a \le x \le b] : a, b \in \mathbb{R}\}$.

Lower bound ($\mathrm{VCdim} \ge 2$): pick two points $x_1 < x_2$. All four labelings are achievable:

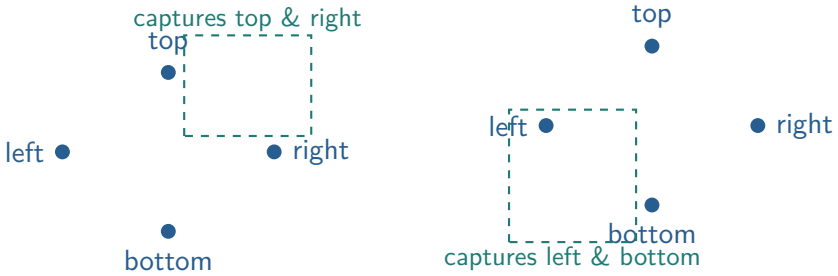
$$00, \quad 10, \quad 01, \quad 11.$$

Upper bound ($\mathrm{VCdim} \le 2$): for any three ordered points $x_1 < x_2 < x_3$, the labeling $1, 0, 1$ is impossible — an interval can't skip the middle point.

Hence $\mathrm{VCdim}(\mathcal{H}) = 2$.

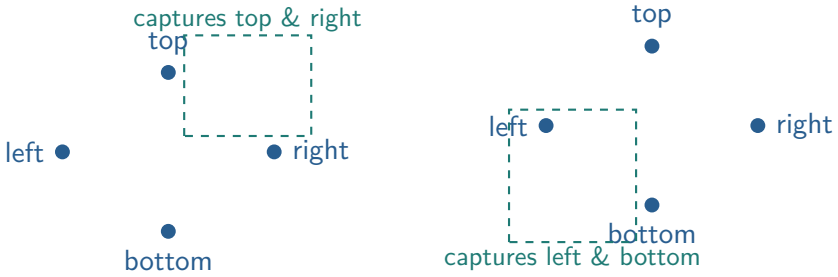
VC example: rectangles — lower bound

$\mathcal{H}$ = axis-aligned rectangles in $\mathbb{R}^2$. Place 4 points in a cross:



VC example: rectangles — lower bound

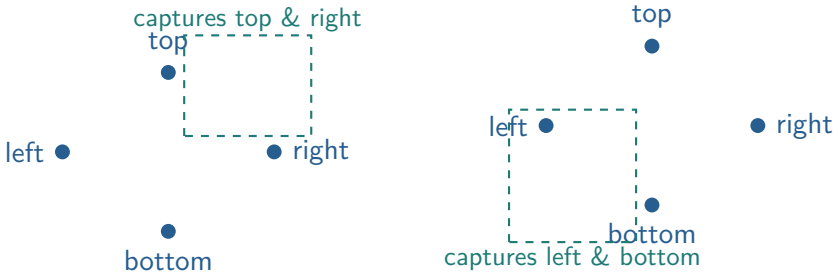
$\mathcal{H}$ = axis-aligned rectangles in $\mathbb{R}^2$. Place 4 points in a cross:



Each boundary edge independently controls one point. For any subset labeled 1, we can draw a rectangle containing exactly those points.

VC example: rectangles — lower bound

$\mathcal{H}$ = axis-aligned rectangles in $\mathbb{R}^2$. Place 4 points in a cross:

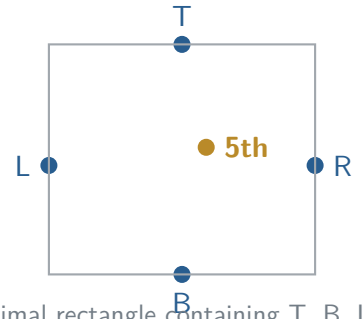


Each boundary edge independently controls one point. For any subset labeled 1, we can draw a rectangle containing exactly those points.

So these 4 points are shattered: $\mathrm{VCdim}(\mathcal{H}) \geq 4$.

VC example: rectangles — upper bound

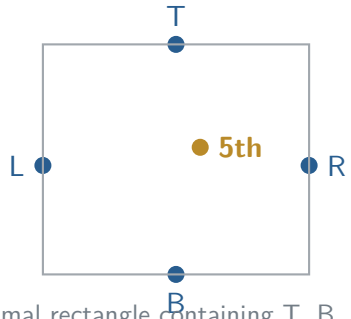
Given any 5 points, identify the four extremals (topmost, bottommost, leftmost, rightmost):



minimal rectangle containing T, B, L, R

VC example: rectangles — upper bound

Given any 5 points, identify the four extremals (topmost, bottommost, leftmost, rightmost):

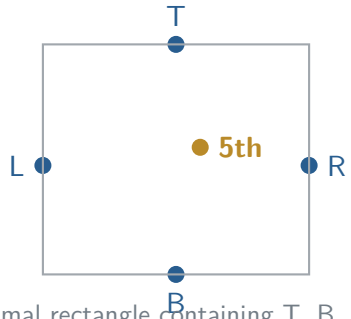


minimal rectangle containing T, B, L, R

The 5th point lies between the extremes in both coordinates — it is inside *every* rectangle that contains the other four.

VC example: rectangles — upper bound

Given any 5 points, identify the four extremals (topmost, bottommost, leftmost, rightmost):



minimal rectangle containing T, B, L, R

The 5th point lies between the extremes in both coordinates — it is inside *every* rectangle that contains the other four.

VC of rectangles: summary

- **Lower bound:** 4 points in a cross can be shattered — each boundary edge controls one point.

VC of rectangles: summary

- **Lower bound:** 4 points in a cross can be shattered — each boundary edge controls one point.
- **Upper bound:** any 5th point is trapped inside the bounding rectangle of the four extremals.

VC of rectangles: summary

- **Lower bound:** 4 points in a cross can be shattered — each boundary edge controls one point.
- **Upper bound:** any 5th point is trapped inside the bounding rectangle of the four extremals.

Hence $\mathrm{VCdim}(\mathcal{H}) = 4$.

VC example: halfspaces in $\mathbb{R}^d$

Consider homogeneous halfspaces

$$\mathcal{H} = \{x \mapsto \mathbf{1}[\langle w, x \rangle \geq 0] : w \in \mathbb{R}^d\}.$$

VC example: halfspaces in $\mathbb{R}^d$

Consider homogeneous halfspaces

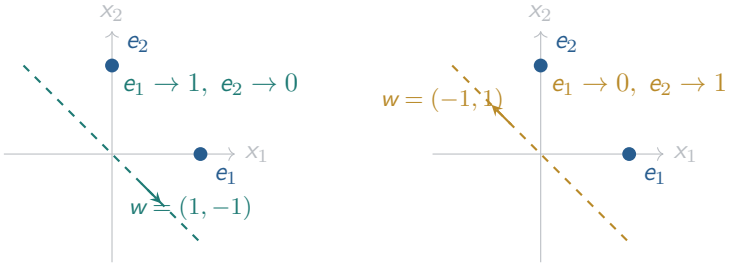
$$\mathcal{H} = \{x \mapsto \mathbf{1}[\langle w, x \rangle \geq 0] : w \in \mathbb{R}^d\}.$$

Lower bound ($\mathrm{VCdim} \geq d$): the standard basis $e_1, \ldots, e_d$ can be shattered.

For any target labeling, choose w so that $\langle w, e_i \rangle = w_i$ has the desired sign.

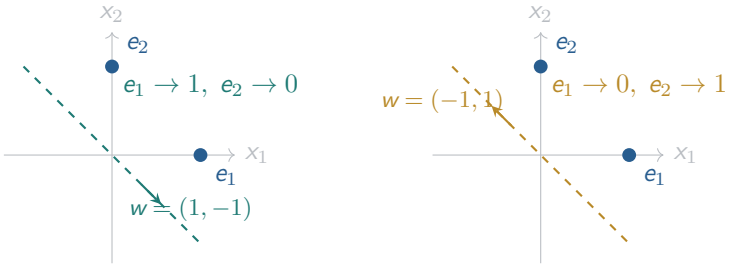
Halfspaces: lower bound illustrated ($\mathbb{R}^2$)

Dashed line $=$ boundary $\langle w, x \rangle = 0$. Arrow w points to the positive side ($\to 1$).



Halfspaces: lower bound illustrated ($\mathbb{R}^2$)

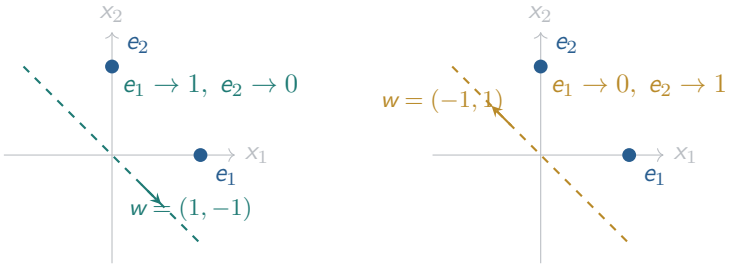
Dashed line = boundary $\langle w, x \rangle = 0$. Arrow w points to the positive side ($\to 1$).



$w = (1, 1)$ and $(-1, -1)$ give the other two labelings — all $2^2 = 4$ patterns achieved.

Halfspaces: lower bound illustrated ($\mathbb{R}^2$)

Dashed line = boundary $\langle w, x \rangle = 0$. Arrow w points to the positive side ($\to 1$).

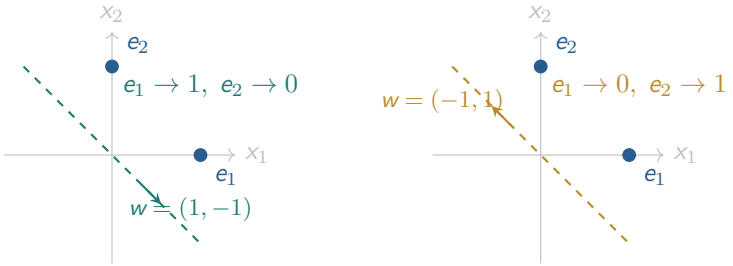


$w = (1, 1)$ and $(-1, -1)$ give the other two labelings — all $2^2 = 4$ patterns achieved.

Each component of w independently controls one basis vector.

Halfspaces: lower bound illustrated ($\mathbb{R}^2$)

Dashed line = boundary $\langle w, x \rangle = 0$. Arrow w points to the positive side ($\to 1$).



$w = (1, 1)$ and $(-1, -1)$ give the other two labelings — all $2^2 = 4$ patterns achieved.

Each component of w independently controls one basis vector.

In $\mathbb{R}^d$: same idea with d basis vectors. So $\mathrm{VCdim}(\mathcal{H}) \geq d$.

VC example: halfspaces — upper bound

Claim: no $d+1$ points in $\mathbb{R}^d$ can be shattered.

VC example: halfspaces — upper bound

Claim: no $d+1$ points in $\mathbb{R}^d$ can be shattered.

Why? Any $d+1$ vectors in $\mathbb{R}^d$ are linearly dependent: there exist α_i, not all zero, such that

$$\alpha_1 x_1 + \cdots + \alpha_{d+1} x_{d+1} = 0.$$

VC example: halfspaces — upper bound

Claim: no $d+1$ points in $\mathbb{R}^d$ can be shattered.

Why? Any $d+1$ vectors in $\mathbb{R}^d$ are linearly dependent: there exist α_i, not all zero, such that

$$\alpha_1 x_1 + \cdots + \alpha_{d+1} x_{d+1} = 0.$$

Assume some $\alpha_i > 0$ and some $\alpha_j < 0$ (the other case is similar).

VC example: halfspaces — upper bound

Claim: no $d+1$ points in $\mathbb{R}^d$ can be shattered.

Why? Any $d+1$ vectors in $\mathbb{R}^d$ are linearly dependent: there exist α_i, not all zero, such that

$$\alpha_1 x_1 + \cdots + \alpha_{d+1} x_{d+1} = 0.$$

Assume some $\alpha_i > 0$ and some $\alpha_j < 0$ (the other case is similar).

Consider the labeling $y_i = \mathbf{1}[\alpha_i > 0]$. If w realizes it, then $\alpha_i \langle w, x_i \rangle \geq 0$ for all i, with strict inequality when $\alpha_i < 0$. So

$$0 = \sum_i \alpha_i \langle w, x_i \rangle > 0. \qquad \text{Contradiction.}$$

VC example: halfspaces — upper bound

Claim: no $d+1$ points in $\mathbb{R}^d$ can be shattered.

Why? Any $d+1$ vectors in $\mathbb{R}^d$ are linearly dependent: there exist α_i, not all zero, such that

$$\alpha_1 x_1 + \cdots + \alpha_{d+1} x_{d+1} = 0.$$

Assume some $\alpha_i > 0$ and some $\alpha_j < 0$ (the other case is similar).

Consider the labeling $y_i = \mathbf{1}[\alpha_i > 0]$. If w realizes it, then $\alpha_i \langle w, x_i \rangle \geq 0$ for all i, with strict inequality when $\alpha_i < 0$. So

$$0 = \sum_i \alpha_i \langle w, x_i \rangle > 0. \qquad \text{Contradiction.}$$

Hence $\mathrm{VCdim}(\mathcal{H}) = d$.

The pattern in every VC computation

Every example needed two separate arguments:

Lower bound

Exhibit d points that *can* be shattered.

The pattern in every VC computation

Every example needed two separate arguments:

Lower bound

Exhibit d points that *can* be shattered.

Upper bound

Show that no set of size $d + 1$ can be shattered.

The pattern in every VC computation

Every example needed two separate arguments:

Lower bound

Exhibit d points that *can* be shattered.

Upper bound

Show that no set of size $d + 1$ can be shattered.

The upper bound is usually harder. One forbidden labeling is enough.

What VC dimension is really measuring

We've computed VCdim for several classes.

Class	VCdim
Thresholds	1
Intervals	2
Rectangles	4
Halfspaces ($\mathbb{R}^d$)	d
All functions	∞

What VC dimension is really measuring

We've computed VCdim for several classes.

What does this number capture?

Class	VCdim
Thresholds	1
Intervals	2
Rectangles	4
Halfspaces ($\mathbb{R}^d$)	d
All functions	∞

What VC dimension is really measuring

We've computed VCdim for several classes.

What does this number capture?

The largest pool size on which the class can still realize *all* labelings — giving the learner no advantage.

Class	VCdim
Thresholds	1
Intervals	2
Rectangles	4
Halfspaces ($\mathbb{R}^d$)	d
All functions	∞

What happens above VCdim?

Below VCdim, the adversary can still force all 2^n labelings — mistake bound $= n$.

What happens above VCdim?

Below VCdim, the adversary can still force all 2^n labelings — mistake bound $= n$.

Above it, some labelings are always impossible — but how many?

What happens above VCdim?

Below VCdim, the adversary can still force all 2^n labelings — mistake bound $= n$.

Above it, some labelings are always impossible — but how many?

$\Gamma_{\mathcal{H}}(n)$ measures the worst-case number of patterns on a pool of size n.

What happens above VCdim?

Below VCdim, the adversary can still force all 2^n labelings — mistake bound $= n$.

Above it, some labelings are always impossible — but how many?

$\Gamma_{\mathcal{H}}(n)$ measures the worst-case number of patterns on a pool of size n.

- If $\Gamma_{\mathcal{H}}(n)$ stays close to 2^n: mistake bound $\approx n$ — barely any advantage.

What happens above VCdim?

Below VCdim, the adversary can still force all 2^n labelings — mistake bound $= n$.

Above it, some labelings are always impossible — but how many?

$\Gamma_{\mathcal{H}}(n)$ measures the worst-case number of patterns on a pool of size n.

- If $\Gamma_{\mathcal{H}}(n)$ stays close to 2^n: mistake bound $\approx n$ — barely any advantage.
- If $\Gamma_{\mathcal{H}}(n)$ drops to polynomial: mistake bound $\ll n$ — dramatically better.

What happens above VCdim?

Below VCdim, the adversary can still force all 2^n labelings — mistake bound $= n$.

Above it, some labelings are always impossible — but how many?

$\Gamma_{\mathcal{H}}(n)$ measures the worst-case number of patterns on a pool of size n.

- If $\Gamma_{\mathcal{H}}(n)$ stays close to 2^n: mistake bound $\approx n$ — barely any advantage.
- If $\Gamma_{\mathcal{H}}(n)$ drops to polynomial: mistake bound $\ll n$ — dramatically better.

Which is it?

Sauer–Shelah lemma

The key counting theorem behind VC theory

Growth beyond the VC dimension: examples

What does $\Gamma_{\mathcal{H}}(n)$ look like for $n > d$?

Growth beyond the VC dimension: examples

What does $\Gamma_{\mathcal{H}}(n)$ look like for $n > d$?

- Thresholds ($d = 1$): $\Gamma_{\mathcal{H}}(n) = n + 1$ — linear in n.

Growth beyond the VC dimension: examples

What does $\Gamma_{\mathcal{H}}(n)$ look like for $n > d$?

- Thresholds ($d = 1$): $\Gamma_{\mathcal{H}}(n) = n + 1$ — linear in n.
- Intervals ($d = 2$): $\Gamma_{\mathcal{H}}(n) = 1 + \binom{n+1}{2} \sim n^2/2$ — quadratic in n.

Growth beyond the VC dimension: examples

What does $\Gamma_{\mathcal{H}}(n)$ look like for $n > d$?

- Thresholds ($d = 1$): $\Gamma_{\mathcal{H}}(n) = n + 1$ — linear in n.
- Intervals ($d = 2$): $\Gamma_{\mathcal{H}}(n) = 1 + \binom{n+1}{2} \sim n^2/2$ — quadratic in n.

Both polynomial, not exponential. Notice the exponents:

- $d = 1$: growth $\sim n^1$

Growth beyond the VC dimension: examples

What does $\Gamma_{\mathcal{H}}(n)$ look like for $n > d$?

- Thresholds ($d = 1$): $\Gamma_{\mathcal{H}}(n) = n + 1$ — linear in n.
- Intervals ($d = 2$): $\Gamma_{\mathcal{H}}(n) = 1 + \binom{n+1}{2} \sim n^2/2$ — quadratic in n.

Both polynomial, not exponential. Notice the exponents:

- $d = 1$: growth $\sim n^1$
- $d = 2$: growth $\sim n^2$

Growth beyond the VC dimension: examples

What does $\Gamma_{\mathcal{H}}(n)$ look like for $n > d$?

- Thresholds ($d = 1$): $\Gamma_{\mathcal{H}}(n) = n + 1$ — linear in n.
- Intervals ($d = 2$): $\Gamma_{\mathcal{H}}(n) = 1 + \binom{n+1}{2} \sim n^2/2$ — quadratic in n.

Both polynomial, not exponential. Notice the exponents:

- $d = 1$: growth $\sim n^1$
- $d = 2$: growth $\sim n^2$

Does $\mathrm{VCdim}(\mathcal{H}) = d$ always force $\Gamma_{\mathcal{H}}(n) = O(n^d)$?

Sauer–Shelah lemma (1972)

Yes:

Sauer–Shelah lemma

If $\mathrm{VCdim}(\mathcal{H}) = d < \infty$, then for every n,

$$\Gamma_{\mathcal{H}}(n) \leq \sum_{k=0}^{d} \binom{n}{k}.$$

In particular, for $n \geq d \geq 1$,

$$\Gamma_{\mathcal{H}}(n) \leq \left(\frac{en}{d}\right)^{d}.$$

Sauer–Shelah lemma (1972)

Yes:

Sauer–Shelah lemma

If $\mathrm{VCdim}(\mathcal{H}) = d < \infty$, then for every n,

$$\Gamma_{\mathcal{H}}(n) \leq \sum_{k=0}^{d} \binom{n}{k}.$$

In particular, for $n \geq d \geq 1$,

$$\Gamma_{\mathcal{H}}(n) \leq \left(\frac{en}{d}\right)^{d}.$$

The bound is $O(n^d)$ — polynomial in n with exponent equal to the VC dimension.

Sauer–Shelah lemma (1972)

Yes:

Sauer–Shelah lemma

If $\mathrm{VCdim}(\mathcal{H}) = d < \infty$, then for every n,

$$\Gamma_{\mathcal{H}}(n) \leq \sum_{k=0}^{d} \binom{n}{k}.$$

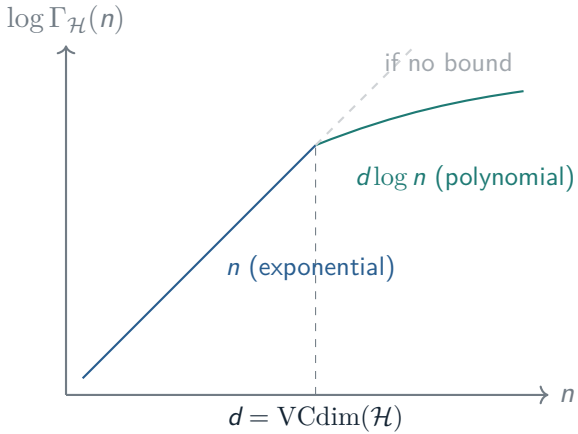
In particular, for $n \geq d \geq 1$,

$$\Gamma_{\mathcal{H}}(n) \leq \left(\frac{en}{d}\right)^{d}.$$

The bound is $O(n^d)$ — polynomial in n with exponent equal to the VC dimension.

So once the class stops shattering, growth can **never** be exponential.

The phase transition in growth



At $n = d$, growth transitions from exponential to polynomial — a sharp phase transition.

Setting up the proof

Notation. Write $\Gamma_d(n)$ for the maximum of $\Gamma_{\mathcal{H}}(n)$ over all classes $\mathcal{H}$ with $\mathrm{VCdim}(\mathcal{H}) \leq d$.

Setting up the proof

Notation. Write $\Gamma_d(n)$ for the maximum of $\Gamma_{\mathcal{H}}(n)$ over all classes $\mathcal{H}$ with $\mathrm{VCdim}(\mathcal{H}) \leq d$.

Goal: prove that for all $n \geq 1$ and $d \geq 0$,

$$\Gamma_d(n) \leq \sum_{k=0}^{d} \binom{n}{k}.$$

Setting up the proof

Notation. Write $\Gamma_d(n)$ for the maximum of $\Gamma_{\mathcal{H}}(n)$ over all classes $\mathcal{H}$ with $\mathrm{VCdim}(\mathcal{H}) \leq d$.

Goal: prove that for all $n \geq 1$ and $d \geq 0$,

$$\Gamma_d(n) \leq \sum_{k=0}^{d} \binom{n}{k}.$$

Strategy: induction on n.

Setting up the proof

Notation. Write $\Gamma_d(n)$ for the maximum of $\Gamma_{\mathcal{H}}(n)$ over all classes $\mathcal{H}$ with $\mathrm{VCdim}(\mathcal{H}) \le d$.

Goal: prove that for all $n \ge 1$ and $d \ge 0$,

$$\Gamma_d(n) \le \sum_{k=0}^{d} \binom{n}{k}.$$

Strategy: induction on n.

Base cases:

- $d = 0$: no single point is shattered, so every pool gets at most 1 labeling. $\Gamma_0(n) = 1 = \binom{n}{0}$.
- $n \le d$: trivially $\Gamma_d(n) \le 2^n = \sum_{k=0}^{d} \binom{n}{k}$.

Inductive step: the prefix idea

Fix $\mathcal{H}$ with $\mathrm{VCdim}(\mathcal{H}) \leq d$ and a pool $C = \{x_1, \ldots, x_n\}$.

Inductive step: the prefix idea

Fix $\mathcal{H}$ with $\mathrm{VCdim}(\mathcal{H}) \leq d$ and a pool $C = \{x_1, \ldots, x_n\}$.

Every labeling $(y_1, \ldots, y_n) \in \mathcal{H}|_C$ has a **prefix**:

$$(y_1, \ldots, y_{n-1}) \quad \text{on } C' = \{x_1, \ldots, x_{n-1}\}.$$

Inductive step: the prefix idea

Fix $\mathcal{H}$ with $\mathrm{VCdim}(\mathcal{H}) \leq d$ and a pool $C = \{x_1, \ldots, x_n\}$.

Every labeling $(y_1, \ldots, y_n) \in \mathcal{H}|_C$ has a **prefix**:

$$(y_1, \ldots, y_{n-1}) \quad \text{on } C' = \{x_1, \ldots, x_{n-1}\}.$$

Some prefixes pin down the label at x_n; others leave it free.

Inductive step: the prefix idea

Fix $\mathcal{H}$ with $\mathrm{VCdim}(\mathcal{H}) \leq d$ and a pool $C = \{x_1, \ldots, x_n\}$.

Every labeling $(y_1, \ldots, y_n) \in \mathcal{H}|_C$ has a **prefix**:

$$(y_1, \ldots, y_{n-1}) \quad \text{on } C' = \{x_1, \ldots, x_{n-1}\}.$$

Some prefixes pin down the label at x_n; others leave it free.

- **Type A:** only one label at x_n is compatible — extends to 1 labeling on C.
- **Type B:** both labels at x_n are compatible — extends to 2 labelings on C.

Example: prefixes for thresholds on $\{1, 2, 3\}$

Thresholds $\mathcal{H} = \{x \mapsto \mathbf{1}[x \le \theta] : \theta \in \mathbb{R}\}$ on $C = \{1, 2, 3\}$:

Example: prefixes for thresholds on $\{1, 2, 3\}$

Thresholds $\mathcal{H} = \{x \mapsto \mathbf{1}[x \le \theta] : \theta \in \mathbb{R}\}$ on $C = \{1, 2, 3\}$:

$x_1 = 1$	$x_2 = 2$	$x_3 = 3$	prefix on $\{x_1, x_2\}$
0	0	0	$(0, 0)$
1	0	0	$(1, 0)$
1	1	0	$(1, 1)$
1	1	1	$(1, 1)$

Example: prefixes for thresholds on $\{1, 2, 3\}$

Thresholds $\mathcal{H} = \{x \mapsto \mathbf{1}[x \le \theta] : \theta \in \mathbb{R}\}$ on $C = \{1, 2, 3\}$:

$x_1 = 1$	$x_2 = 2$	$x_3 = 3$	prefix on $\{x_1, x_2\}$
0	0	0	$(0, 0)$
1	0	0	$(1, 0)$
1	1	0	$(1, 1)$
1	1	1	$(1, 1)$

Three distinct prefixes (recall: 1s are on the left, 0s on the right):

- $(0, 0)$: $\theta < 1$, so x_3 must also be $0 \rightarrow$ **Type A**
- $(1, 0)$: $\theta \in [1, 2)$, so x_3 must be $0 \rightarrow$ **Type A**
- $(1, 1)$: $\theta \ge 2$, but x_3 depends on whether $\theta < 3$ or $\theta \ge 3 \rightarrow$ **Type B**

Example: prefixes for thresholds on $\{1, 2, 3\}$

Thresholds $\mathcal{H} = \{x \mapsto \mathbf{1}[x \le \theta] : \theta \in \mathbb{R}\}$ on $C = \{1, 2, 3\}$:

$x_1 = 1$	$x_2 = 2$	$x_3 = 3$	prefix on $\{x_1, x_2\}$
0	0	0	$(0, 0)$
1	0	0	$(1, 0)$
1	1	0	$(1, 1)$
1	1	1	$(1, 1)$

Three distinct prefixes (recall: 1s are on the left, 0s on the right):

- $(0, 0)$: $\theta < 1$, so x_3 must also be $0 \to$ **Type A**
- $(1, 0)$: $\theta \in [1, 2)$, so x_3 must be $0 \to$ **Type A**
- $(1, 1)$: $\theta \ge 2$, but x_3 depends on whether $\theta < 3$ or $\theta \ge 3 \to$ **Type B**

Total labelings: $4 = \underbrace{3}_{\text{prefixes}} + \underbrace{1}_{\text{Type B}}$.

Counting with the split

Write a = number of Type A prefixes, b = number of Type B prefixes.
Every prefix is either Type A or Type B, so $a + b$ = total distinct prefixes.

Counting with the split

Write $a =$ number of Type A prefixes, $b =$ number of Type B prefixes.
Every prefix is either Type A or Type B, so $a + b =$ total distinct prefixes.

Each Type A prefix contributes 1 labeling; each Type B contributes 2:

$$\Gamma_{\mathcal{H}}(C) = a + 2b = \underbrace{(a+b)}_{\text{distinct prefixes}} + \underbrace{b}_{\text{extra from Type B}}.$$

Counting with the split

Write $a =$ number of Type A prefixes, $b =$ number of Type B prefixes.
Every prefix is either Type A or Type B, so $a + b =$ total distinct prefixes.

Each Type A prefix contributes 1 labeling; each Type B contributes 2:

$$\Gamma_{\mathcal{H}}(C) = a + 2b = \underbrace{(a+b)}_{\text{distinct prefixes}} + \underbrace{b}_{\text{extra from Type B}}.$$

- $a + b$: the number of distinct prefixes. Each prefix is a labeling that some $h \in \mathcal{H}$ produces on $C' = \{x_1, \ldots, x_{n-1}\}$. So $a + b$ counts distinct labelings of $\mathcal{H}$ on C'. Since $\mathrm{VCdim}(\mathcal{H}) \leq d$, we get $a + b \leq \Gamma_d(n-1)$.

Counting with the split

Write a = number of Type A prefixes, b = number of Type B prefixes.
Every prefix is either Type A or Type B, so $a + b$ = total distinct prefixes.

Each Type A prefix contributes 1 labeling; each Type B contributes 2:

$$\Gamma_{\mathcal{H}}(C) = a + 2b = \underbrace{(a+b)}_{\text{distinct prefixes}} + \underbrace{b}_{\text{extra from Type B}}.$$

- $a + b$: the number of distinct prefixes. Each prefix is a labeling that some $h \in \mathcal{H}$ produces on $C' = \{x_1, \ldots, x_{n-1}\}$. So $a + b$ counts distinct labelings of $\mathcal{H}$ on C'. Since $\mathrm{VCdim}(\mathcal{H}) \leq d$, we get $a + b \leq \Gamma_d(n-1)$.
- b: the number of extra labelings that x_n creates. For each Type B prefix, both labels at x_n are possible, so one prefix produces two labelings on C instead of one. How large can b be?

Counting with the split

Write a = number of Type A prefixes, b = number of Type B prefixes.
Every prefix is either Type A or Type B, so $a + b$ = total distinct prefixes.

Each Type A prefix contributes 1 labeling; each Type B contributes 2:

$$\Gamma_{\mathcal{H}}(C) = a + 2b = \underbrace{(a+b)}_{\text{distinct prefixes}} + \underbrace{b}_{\text{extra from Type B}}.$$

- $a + b$: the number of distinct prefixes. Each prefix is a labeling that some $h \in \mathcal{H}$ produces on $C' = \{x_1, \ldots, x_{n-1}\}$. So $a + b$ counts distinct labelings of $\mathcal{H}$ on C'. Since $\mathrm{VCdim}(\mathcal{H}) \leq d$, we get $a + b \leq \Gamma_d(n-1)$.
- b: the number of extra labelings that x_n creates. For each Type B prefix, both labels at x_n are possible, so one prefix produces two labelings on C instead of one. How large can b be?

Therefore $\Gamma_{\mathcal{H}}(C) \leq \Gamma_d(n-1) + b$. Bounding b is where the VC dimension enters.

Why Type B prefixes have lower VC dimension

Claim: the Type B prefixes can shatter at most $d - 1$ points among $\{x_1, \ldots, x_{n-1}\}$.

Why Type B prefixes have lower VC dimension

Claim: the Type B prefixes can shatter at most $d - 1$ points among $\{x_1, \ldots, x_{n-1}\}$.

Proof by contradiction. Suppose the Type B prefixes shatter d points $\{x_{i_1}, \ldots, x_{i_d}\} \subseteq \{x_1, \ldots, x_{n-1}\}$.

Why Type B prefixes have lower VC dimension

Claim: the Type B prefixes can shatter at most $d-1$ points among $\{x_1, \ldots, x_{n-1}\}$.

Proof by contradiction. Suppose the Type B prefixes shatter d points $\{x_{i_1}, \ldots, x_{i_d}\} \subseteq \{x_1, \ldots, x_{n-1}\}$.

Then for every labeling of these d points, some Type B prefix realizes it.

Why Type B prefixes have lower VC dimension

Claim: the Type B prefixes can shatter at most $d-1$ points among $\{x_1, \ldots, x_{n-1}\}$.

Proof by contradiction. Suppose the Type B prefixes shatter d points $\{x_{i_1}, \ldots, x_{i_d}\} \subseteq \{x_1, \ldots, x_{n-1}\}$.

Then for every labeling of these d points, some Type B prefix realizes it.

But Type B means both extensions at x_n are realized. So every labeling of $\{x_{i_1}, \ldots, x_{i_d}\}$ *and* every label at x_n is realized by some $h \in \mathcal{H}$.

Why Type B prefixes have lower VC dimension

Claim: the Type B prefixes can shatter at most $d-1$ points among $\{x_1, \ldots, x_{n-1}\}$.

Proof by contradiction. Suppose the Type B prefixes shatter d points $\{x_{i_1}, \ldots, x_{i_d}\} \subseteq \{x_1, \ldots, x_{n-1}\}$.

Then for every labeling of these d points, some Type B prefix realizes it.

But Type B means both extensions at x_n are realized. So every labeling of $\{x_{i_1}, \ldots, x_{i_d}\}$ *and* every label at x_n is realized by some $h \in \mathcal{H}$.

That means $\mathcal{H}$ shatters $\{x_{i_1}, \ldots, x_{i_d}, x_n\}$ — a set of $d+1$ points.
Contradiction with $\mathrm{VCdim}(\mathcal{H}) \leq d$.

Assembling and solving the recurrence

From the prefix counting: $\Gamma_{\mathcal{H}}(C) \leq \Gamma_d(n-1) + b$.

Assembling and solving the recurrence

From the prefix counting: $\Gamma_{\mathcal{H}}(C) \leq \Gamma_d(n-1) + b$.

Type B prefixes shatter at most $d-1$ points, so $b \leq \Gamma_{d-1}(n-1)$.

Assembling and solving the recurrence

From the prefix counting: $\Gamma_{\mathcal{H}}(C) \leq \Gamma_d(n-1) + b$.

Type B prefixes shatter at most $d-1$ points, so $b \leq \Gamma_{d-1}(n-1)$.

Since this holds for every pool C of size n and every $\mathcal{H}$ with $\mathrm{VCdim}(\mathcal{H}) \leq d$:

$$\boxed{\Gamma_d(n) \leq \Gamma_d(n-1) + \Gamma_{d-1}(n-1)}$$

Assembling and solving the recurrence

From the prefix counting: $\Gamma_{\mathcal{H}}(C) \le \Gamma_d(n-1) + b$.

Type B prefixes shatter at most $d-1$ points, so $b \le \Gamma_{d-1}(n-1)$.

Since this holds for every pool C of size n and every $\mathcal{H}$ with $\mathrm{VCdim}(\mathcal{H}) \le d$:

$$\boxed{\Gamma_d(n) \le \Gamma_d(n-1) + \Gamma_{d-1}(n-1)}$$

Base cases: $\Gamma_0(n) = 1$, $\Gamma_d(n) = 2^n$ for $n \le d$.
This is Pascal's rule, so by induction on n:

$$\Gamma_d(n) \le \sum_{k=0}^{d} \binom{n}{k}.$$

In particular, for $n \ge d \ge 1$,

$$\Gamma_d(n) \le \left(\frac{en}{d}\right)^d.$$

Sauer–Shelah: recap

Setup. $\Gamma_{\mathcal{H}}(n) = \max_{|C|=n}$ number of distinct labelings $\mathcal{H}$ produces on C.

Sauer–Shelah: recap

Setup. $\Gamma_{\mathcal{H}}(n) = \max_{|C|=n}$ number of distinct labelings $\mathcal{H}$ produces on C.

$\mathrm{VCdim}(\mathcal{H}) = d$: the largest pool size that $\mathcal{H}$ can shatter.

Sauer–Shelah: recap

Setup. $\Gamma_{\mathcal{H}}(n) = \max_{|C|=n}$ number of distinct labelings $\mathcal{H}$ produces on C.

$\mathrm{VCdim}(\mathcal{H}) = d$: the largest pool size that $\mathcal{H}$ can shatter.

Sauer–Shelah Lemma

For any class $\mathcal{H}$ with $\mathrm{VCdim}(\mathcal{H}) = d$:

$$\Gamma_{\mathcal{H}}(n) \leq \sum_{k=0}^{d} \binom{n}{k}.$$

In particular, for $n \geq d \geq 1$,

$$\Gamma_{\mathcal{H}}(n) \leq \left(\frac{en}{d}\right)^{d}.$$

Sauer–Shelah: recap

Setup. $\Gamma_{\mathcal{H}}(n) = \max_{|C|=n}$ number of distinct labelings $\mathcal{H}$ produces on C.

$\mathrm{VCdim}(\mathcal{H}) = d$: the largest pool size that $\mathcal{H}$ can shatter.

Sauer–Shelah Lemma

For any class $\mathcal{H}$ with $\mathrm{VCdim}(\mathcal{H}) = d$:

$$\Gamma_{\mathcal{H}}(n) \leq \sum_{k=0}^{d} \binom{n}{k}.$$

In particular, for $n \geq d \geq 1$,

$$\Gamma_{\mathcal{H}}(n) \leq \left(\frac{en}{d}\right)^{d}.$$

Once the pool size n exceeds d, growth is polynomial in n, not exponential.

What this means for learning

Recall: Halving on a pool of size n makes at most $\log_2 \Gamma_{\mathcal{H}}(n)$ mistakes.

What this means for learning

Recall: Halving on a pool of size n makes at most $\log_2 \Gamma_{\mathcal{H}}(n)$ mistakes.

Applying Sauer–Shelah, for $n \geq d \geq 1$:

$$\text{mistakes} \leq \log_2 \left(\frac{en}{d}\right)^d = d \log_2 \frac{en}{d} = O(d \log n).$$

What this means for learning

Recall: Halving on a pool of size n makes at most $\log_2 \Gamma_{\mathcal{H}}(n)$ mistakes.

Applying Sauer–Shelah, for $n \geq d \geq 1$:

$$\text{mistakes} \leq \log_2 \left(\frac{en}{d}\right)^d = d \log_2 \frac{en}{d} = O(d \log n).$$

The mistake bound depends on the pool size n only logarithmically.

What this means for learning

Recall: Halving on a pool of size n makes at most $\log_2 \Gamma_{\mathcal{H}}(n)$ mistakes.

Applying Sauer–Shelah, for $n \geq d \geq 1$:

$$\text{mistakes} \leq \log_2 \left(\frac{en}{d}\right)^d = d \log_2 \frac{en}{d} = O(d \log n).$$

The mistake bound depends on the pool size n only logarithmically.

The dominant factor is $d = \mathrm{VCdim}(\mathcal{H})$: the simpler the class, the fewer mistakes.

The story of Week 2

Question: what controls learning beyond $|\mathcal{H}|$?

The story of Week 2

Question: what controls learning beyond $|\mathcal{H}|$?

1. On finite data, what matters is how many labelings $\mathcal{H}$ can produce.

The story of Week 2

Question: what controls learning beyond $|\mathcal{H}|$?

1. On finite data, what matters is how many labelings $\mathcal{H}$ can produce.
2. The growth function $\Gamma_{\mathcal{H}}(n)$ measures this worst-case richness.

The story of Week 2

Question: what controls learning beyond $|\mathcal{H}|$?

1. On finite data, what matters is how many labelings $\mathcal{H}$ can produce.
2. The growth function $\Gamma_{\mathcal{H}}(n)$ measures this worst-case richness.
3. VC dimension identifies where that richness must break.

The story of Week 2

Question: what controls learning beyond $|\mathcal{H}|$?

1. On finite data, what matters is how many labelings $\mathcal{H}$ can produce.
2. The growth function $\Gamma_{\mathcal{H}}(n)$ measures this worst-case richness.
3. VC dimension identifies where that richness must break.
4. Sauer–Shelah: once $n > d$, growth drops from exponential to polynomial.

The story of Week 2

Question: what controls learning beyond $|\mathcal{H}|$?

1. On finite data, what matters is how many labelings $\mathcal{H}$ can produce.
2. The growth function $\Gamma_{\mathcal{H}}(n)$ measures this worst-case richness.
3. VC dimension identifies where that richness must break.
4. Sauer–Shelah: once $n > d$, growth drops from exponential to polynomial.

Payoff. Halving + Sauer–Shelah: $O(d \log n)$ mistakes.
One number d governs learnability.

What's next?

Today we worked in the **online transductive** setting: fixed pool, adversarial reveal order.

What's next?

Today we worked in the **online transductive** setting: fixed pool, adversarial reveal order.

Next steps:

- **Transductive setting:** given a pool, observe labels on a training subset, predict the rest.

What's next?

Today we worked in the **online transductive** setting: fixed pool, adversarial reveal order.

Next steps:

- **Transductive setting:** given a pool, observe labels on a training subset, predict the rest.
- **Statistical (i.i.d.) setting:** data is drawn from an unknown distribution. Does VC dimension still control learning?

Project idea: formalizing the combinatorial core

Everything in this lecture forms a single chain of results:

$$\text{Restriction} \to \text{Growth function} \to \text{Shattering} \to \text{VC dimension} \to \text{Sauer–Shelah}$$

Project idea: formalizing the combinatorial core

Everything in this lecture forms a single chain of results:

$$\text{Restriction} \to \text{Growth function} \to \text{Shattering} \to \text{VC dimension} \to \text{Sauer–Shelah}$$

Project: formalize this chain in Lean 4.

Project idea: formalizing the combinatorial core

Everything in this lecture forms a single chain of results:

$$\text{Restriction} \to \text{Growth function} \to \text{Shattering} \to \text{VC dimension} \to \text{Sauer–Shelah}$$

Project: formalize this chain in Lean 4.

- Define restriction, growth function, shattering, and VC dimension.
- State and prove the Halving mistake bound: $M \leq \log_2 N$.
- State and prove Sauer–Shelah.
- Combine them: $\mathrm{VCdim}(\mathcal{H}) = d \implies M = O(d \log n)$.