Logic, Data Examples and Learning

Lecture 3/5: When do fitting concepts generalize to unseen examples?

Balder ten Cate

Tsinghua University, May 14, 2026

3. When Do Fitting Concepts Generalize?

Recap: Where Lecture 1 Ended

Lecture 1 mainly went in the direction

$$\text{concepts} \longrightarrow \text{examples}.$$

We asked:

- How can examples explain what a concept means?
- Can a finite set of examples identify a concept uniquely?
- How does the answer depend on the concept class?

The final case study was $\mathrm{FO}_{\exists,\wedge,\top}[\mathbf{S}]$:

- it is monotone with respect to the homomorphism preorder;
- unique characterizations led us to frontiers in the homomorphism lattice;
- by a classical theorem of Erdős, frontiers do not always exist.

Recap: What Lecture 2 Added

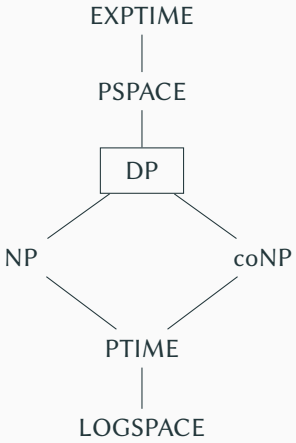
Lecture 2 reversed the direction:

$$\text{labeled examples} \longrightarrow \text{concepts}.$$

We focused on *fitting algorithms*.

- For RECT^{2D}, there is a simple polynomial-time canonical candidate for fitting.
- For richer classes such as U-RECT^{2D}, exact minimum-size fitting is already hard.
- Set Cover gave us our first approximation algorithm for fitting.
- The same themes reappeared for logical concept classes such as $\text{PL}_{\wedge,\top}[\text{PROP}]$ and $\text{FO}_{\exists,\wedge,\top}[\mathbf{S}]$.

Recap: The Larger Complexity Picture



- PTIME sits below NP and coNP.
- PSPACE is a larger class containing many game-like problems (also, the satisfiability problem for modal logics such as **K** and **S4**)
- Lecture 2 mostly lived around PTIME, NP-hardness, and approximation.

Recap: Some Natural Problems Resist Classification

There are famous problems whose exact complexity is still unknown.

Graph Isomorphism

Best known upper bound: quasi-polynomial time.

It is not known to be in PTIME, but it is also not known to be NP-hard.

Truth of existential first-order sentences $\exists x_1 \dots x_n \phi$ over $(\mathbb{R}, +, \cdot, <, 0, 1)$

Known lower bound: NP-hard.

Known upper bound: in PSPACE (in PTIME if n is bounded)

So people write NP $\subseteq \exists\mathbb{R} \subseteq$ PSPACE, where $\exists\mathbb{R}$ is defined as the complexity class of all problems that have a PTIME reduction to this problem (so this problem is by definition $\exists\mathbb{R}$-complete).

Recap: Small Fitting Concepts

A major theme at the end of Lecture 2 was:

can we fit the data using a concept that is nearly as small as possible?

Definition (Occam algorithm)

A fitting algorithm `alg` is an *Occam algorithm* if, on every consistent input E, it outputs a fitting concept of size at most

$$f(\ell_{opt}) \cdot |E|^{\alpha},$$

where ℓ_{opt} is the size of the smallest fitting concept, f is polynomial, and $\alpha < 1$.

Recap: Why This Notion Was Introduced

At first sight, this looks a bit technical.

But the intuition is simple:

- a concept that merely memorizes the sample is often too large;
- an Occam algorithm is forced to *compress* the information in the sample;
- in Lecture 2, this was mainly a size notion;
- today we will see that it also has consequences for generalization.

Recap: Example

For U-RECT^{2D}, the greedy fitting algorithm returns a concept of size at most

$$(\ln n + 1)\ell_{opt},$$

where n is the number of positive examples.

For large n, we have $\ln n + 1 < 2\sqrt{n}$, and therefore

$$(\ln n + 1)\ell_{opt} \leq 2\ell_{opt} \cdot n^{1/2}.$$

So the greedy fitting algorithm for U-RECT^{2D} is an Occam algorithm.

Recap: Where This Left Us

Concept class	Fitting algorithm	PTIME	Occam?
RECT^{2D}	standard fitting algorithm	yes	n/a
U-RECT^{2D}	trivial fitting algorithm	yes	no
U-RECT^{2D}	greedy fitting algorithm	yes	yes

This sets up the question for today:

does being small, or nearly small, help a fitting concept generalize to unseen examples?

What We Will Do Now

Up to now, a fitting concept only had to explain the examples we had already seen.

But usually the examples come from some larger process:

- data generated by nature,
- data collected in an experiment,
- or data sampled from a population.

So now we ask when a fitting concept also makes good predictions on *new* examples.

1. We introduce expected error and the PAC viewpoint.
2. We introduce VC dimension as a measure of richness.
3. We state the theorem linking finite VC dimension to uniform PAC learnability.
4. We return to Occam algorithms and see why they imply PAC learnability.

Probability Refresher

Reminder: What Is a Probability Distribution?

To talk about generalization, we need a way of saying how likely different examples are.

Definition (probability distribution)

A probability distribution tells us how mass is spread over a space of possibilities.

There are two common cases:

- **discrete:** mass sits on individual points and the masses sum to 1;
- **continuous:** mass is spread out by a density whose total integral is 1.

Discrete and Continuous

Discrete

Function $\mathbb{P} : X \to [0, 1]$ so that

$$\sum_{x \in X} \mathbb{P}(x) = 1$$

Works well for countable spaces.

Example: a probability distribution on truth assignments to $\{p_1, \ldots, p_n\}$.

Continuous

Partial function $\mathbb{P} : 2^X \hookrightarrow [0, 1]$ satisfying certain natural requirements (the "Kolmogorov axioms")

Typically needed to model some distributions over uncountable spaces.

Example: the uniform distribution on the square $[0, 1] \times [0, 1]$ in the plane.

The results in this lecture apply to both, but for simplicity you may always think of discrete probability distributions.

Example Distributions

To talk about generalization, we need a probabilistic view of examples.

Definition (example distribution)

An *example distribution* $\mathbb{P}$ for a concept class $\mathsf{C} = (C, \mathcal{X}, \lambda)$ is a probability distribution over the example space $\mathcal{X}$.

Intuitively, $\mathbb{P}$ tells us how likely different examples are to show up in the world.

- Some examples may be common.
- Others may be very rare.
- Good generalization should care more about mistakes on common examples.

Expected Error

Let c_t be the target concept and let c be some hypothesis.

Definition (expected error)

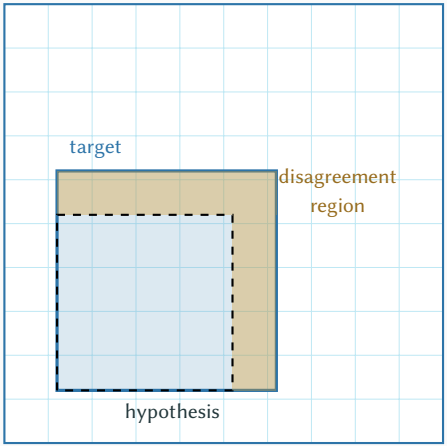
The expected error of c with respect to c_t and $\mathbb{P}$ is

$$\text{error}_{c_t,\mathbb{P}}(c) := \text{Pr}_{e\sim\mathbb{P}}\big(e \in \lambda(c) \oplus \lambda(c_t)\big),$$

i.e. the probability that c and c_t disagree on a random example.

We say that c is ε-*good* if $\text{error}_{c_t,\mathbb{P}}(c) \leq \varepsilon$.

Picture: Error Is Mass of the Disagreement Region



Under the uniform distribution on the ambient square:

- the target concept is the blue rectangle;
- the hypothesis is the dashed black rectangle;
- the orange strip is the disagreement region.

The expected error is exactly the probability mass of that orange region.

Quiz

Quiz 1

In $\mathsf{PL}_{\wedge,\top}[\{p_1, \ldots, p_5\}]$, let

$$c_t = p_1 \wedge p_2 \wedge p_3 \qquad \text{and} \qquad c = p_1 \wedge p_2 \wedge p_3 \wedge p_4.$$

Under the uniform distribution on truth assignments,

$$\mathrm{error}_{c_t,\mathbb{P}}(c) =$$

Quiz

Quiz 1

In $\mathsf{PL}_{\wedge,\top}[\{p_1, \ldots, p_5\}]$, let

$$c_t = p_1 \wedge p_2 \wedge p_3 \qquad \text{and} \qquad c = p_1 \wedge p_2 \wedge p_3 \wedge p_4.$$

Under the uniform distribution on truth assignments,

$$\mathrm{error}_{c_t, \mathbb{P}}(c) = 2/32 = 0.0625.$$

Quiz

Quiz 2

In RECT^{2D}, compare $\mathrm{rect}(1, 10, 1, 10)$ with $\mathrm{rect}(1, 9, 1, 9)$ under the uniform distribution on $[0, 100] \times [0, 100]$).

Quiz

Quiz 2

In RECT^{2D}, compare $\mathrm{rect}(1, 10, 1, 10)$ with $\mathrm{rect}(1, 9, 1, 9)$ under the uniform distribution on $[0, 100] \times [0, 100]$).

The disagreement area is 19, so

$$\mathrm{error}_{c_t, \mathbb{P}}(c) = 19/10000 = 0.0019.$$

PAC Learning

From Fitting to PAC Learning

If the input examples form a large random sample from some example distribution $\mathbb{P}$, we would like the output concept to perform well on *new* examples drawn from the same distribution.

This is formalized by the notion of PAC learning, introduced by Valiant in 1984.

- **P**robably
- **A**pproximately
- **C**orrect

What “PAC” Means

- **approximately correct:** the expected error should be at most ε;
- **probably:** this guarantee should fail with probability at most δ;
- the required sample size is allowed to depend on ε and δ.

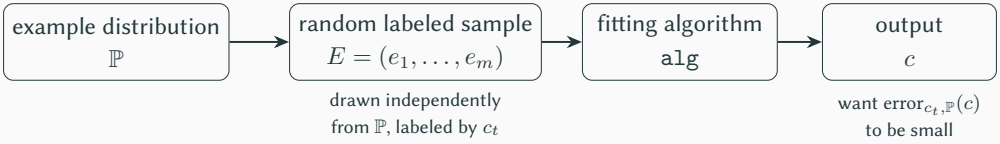
The role of δ is important: even with a good algorithm, we can never rule out the possibility of an unlucky random sample.

Random Labeled Samples

Fix a target concept $c_t \in C$ and an example distribution $\mathbb{P}$.

A *random labeled sample of size* m is obtained by:

1. drawing examples $e_1, \ldots, e_m$ independently from $\mathbb{P}$;
2. labeling each one according to whether it belongs to $\lambda(c_t)$.



This is the setting in which PAC learning is defined.

PAC Property versus Uniform PAC Property

Let `alg` be a fitting algorithm for $\mathsf{C} = (C, \mathcal{X}, \lambda)$.

Uniform PAC property

There exists a function $f(\cdot, \cdot)$ such that for all $\delta, \varepsilon > 0$, all target concepts c_t, and all example distributions $\mathbb{P}$:

if `alg` is run on a random labeled sample of size

$$f(1/\delta, 1/\varepsilon),$$

then with probability at least $1 - \delta$, the output concept c satisfies

$$\text{error}_{c_t, \mathbb{P}}(c) \leq \varepsilon.$$

PAC Property

Let `alg` be a fitting algorithm for $\mathsf{C} = (C, \mathcal{X}, \lambda)$.

PAC property

There exists a function $f(\cdot, \cdot, \cdot)$ such that for all $\delta, \varepsilon > 0$, all target concepts c_t, and all example distributions $\mathbb{P}$:

if `alg` is run on a random labeled sample of size

$$f(1/\delta, 1/\varepsilon, |c_t|),$$

then with probability at least $1 - \delta$, the output concept c satisfies

$$\text{error}_{c_t, \mathbb{P}}(c) \leq \varepsilon.$$

Plain-Language Reading

- **PAC:** with enough random examples, the algorithm usually outputs a low-error concept.
- **Uniform PAC:** the required number of examples depends only on δ and ε, not on the size of the unknown target concept.

If the corresponding sample complexity function is polynomial, we speak of the *polynomial-PAC* or *uniform polynomial-PAC* property.

A Question We Will Come Back To

Which of the following fitting algorithms do you think have the (uniform) PAC property?

1. the standard fitting algorithm for RECT^{2D};
2. the trivial fitting algorithm for U-RECT^{2D};
3. the greedy fitting algorithm for U-RECT^{2D}.

We will answer this after introducing VC dimension.

VC Dimension

Shattering and VC Dimension

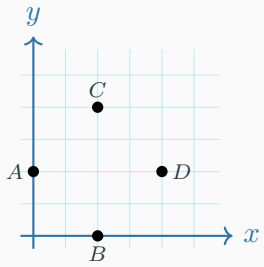
Definition (VC dimension)

Let $\mathsf{C} = (C, \mathcal{X}, \lambda)$ be a concept class.

- A set of unlabeled examples $X \subseteq \mathcal{X}$ is *shattered* by C if every labeling of X is consistent with some concept in C.
- The *VC dimension* of C is the largest size of a shattered set (or ∞ if there is no finite upper bound).

VC dimension measures how rich a concept class is, and how much freedom it has to overfit.

Example: RECT^{2D} Has VC Dimension 4



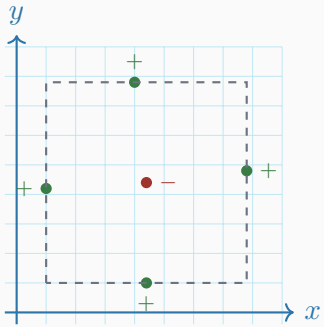
The four points

$$(0,1),\ (1,0),\ (1,2),\ (2,1)$$

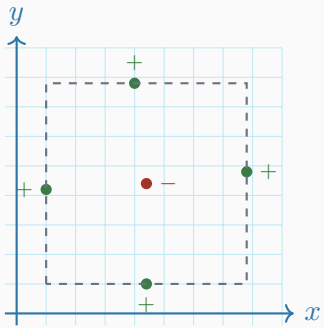
can be shattered by axis-aligned rectangles.

For every choice of positive and negative labels on these four points, there is a rectangle that fits exactly that labeling.

Why No Five-Point Set Can Be Shattered by RECT^{2D}



Why No Five-Point Set Can Be Shattered by RECT^{2D}



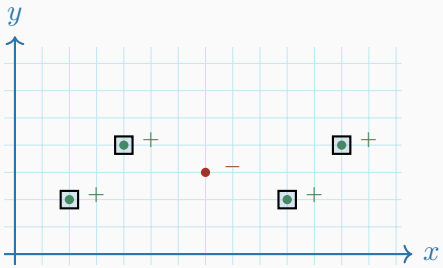
Take any set of five distinct points.
Label as *positive*:

- a point with minimal x-coordinate,
- a point with maximal x-coordinate,
- a point with minimal y-coordinate,
- a point with maximal y-coordinate.

Label the remaining point negative.
Any rectangle containing the four extremal positive points must also contain the negative point.

Therefore, no set of size 5 is shattered, so $\mathrm{VCdim}(\text{RECT}^{2D}) = 4$.

Example: U-RECT^{2D} Has Infinite VC Dimension



Every finite set of points can be shattered by unions of rectangles:

- declare any chosen subset positive;
- put one tiny rectangle around each positive point.

So U-RECT^{2D} has infinite VC dimension.

By contrast, the class U$^{\ell}$RECT^{2D} of unions of at most ℓ rectangles has finite VC dimension, in fact $O(\ell \log \ell)$.

Finite Concept Classes Always Have Finite VC Dimension

Proposition

Let C be a concept class with at most N pairwise non-equivalent concepts. Then

$$\mathrm{VCdim}(\mathsf{C}) \leq \log_2 N.$$

Reason

To shatter a set of size d, we need 2^d different labelings, hence at least 2^d pairwise non-equivalent concepts.

Two Quick Propositional Examples

Quiz 1

What is the VC dimension of $\mathsf{PL}_{\wedge,\top}[\{p_1, \ldots, p_n\}]$?

Answer: n.

Quiz 2

What is the VC dimension of full propositional logic over $\{p_1, \ldots, p_n\}$?

Answer: 2^n, because the example space itself has size 2^n and full propositional logic can realize every subset of it.

Intuition

VC dimension measures the expressive freedom of a concept class.

- If the VC dimension is small, then the class is forced to generalize.
- If the VC dimension is very large, then the class can memorize arbitrary labelings.

Two Fundamental Theorems

Finite Concept Classes

Theorem

Let $\mathsf{C} = (C, \mathcal{X}, \lambda)$ contain at most N pairwise non-equivalent concepts. Let $c_t \in C$, let $\mathbb{P}$ be any example distribution, and let $\delta, \varepsilon > 0$.

If a random labeled sample E has size greater than

$$(1/\varepsilon) \cdot \ln(N/\delta),$$

then with probability at least $1 - \delta$, every concept that fits E has expected error at most ε.

Intuition

A sufficiently large random sample is likely to expose every *bad* concept.

First Consequence

Corollary

If C is finite, then every fitting algorithm for C has the uniform polynomial-PAC property.

This is a reassuring warm-up, but it does not yet solve our running examples:

- RECT^{2D} is infinite;
- U-RECT^{2D} is infinite.

Fundamental Theorem of PAC Learning, Part 1

Theorem

Let C be a concept class of VC dimension d. Let $c_t \in C$, let $\mathbb{P}$ be any example distribution, and let $\delta, \varepsilon > 0$.

If a random labeled sample E has size greater than

$$(4/\varepsilon) \cdot \big(d \log_2(12/\varepsilon) + \log_2(2/\delta)\big),$$

then with probability at least $1 - \delta$, every concept that fits E has expected error at most ε.

Corollary

If a concept class has finite VC dimension, then every fitting algorithm for it has the uniform polynomial-PAC property.

Fundamental Theorem of PAC Learning, Part 2

Theorem

If a concept class has infinite VC dimension, then no fitting algorithm for it has the uniform PAC property.

So for fitting algorithms, the uniform PAC question has a surprisingly clean answer: it depends only on the concept class, not on the specific fitting algorithm.

Back to Our Running Examples

Which fitting algorithms have the *uniform* PAC property?

1. The standard fitting algorithm for RECT^{2D}.
2. The trivial fitting algorithm for U-RECT^{2D}.
3. The greedy fitting algorithm for U-RECT^{2D}.

Answer:

- 1 does, because $\mathrm{VCdim}(\text{RECT}^{2D}) = 4$.
- 2 and 3 do not, because $\mathrm{VCdim}(\text{U-RECT}^{2D}) = \infty$.

Occam Algorithms Are PAC Learners

Theorem

Let C $= (C, \mathcal{X}, \lambda)$ be a concept class such that the subclass C_ℓ of concepts of size at most ℓ has VC dimension

$$O(\ell \log \ell).$$

Then every Occam algorithm for C has the polynomial-PAC property.

In slogan form:

Occam algorithms are PAC learners.

Second Consequence for U-RECT^{2D}

We already know:

- the greedy fitting algorithm for U-RECT^{2D} is an Occam algorithm;
- the classes U$^{\ell}$RECT^{2D} have VC dimension $O(\ell \log \ell)$.

Therefore:

Conclusion

The greedy fitting algorithm for U-RECT^{2D} has the polynomial-PAC property.

So although it does *not* have the uniform PAC property, it still provably generalizes.

Summary

Concept class	Fitting algorithm	PTIME	Occam?	uniform PAC?	PAC?
RECT^{2D}	standard fitting algorithm	yes	n/a	yes	yes
U-RECT^{2D}	trivial fitting algorithm	yes	no	no	no
U-RECT^{2D}	greedy fitting algorithm	yes	yes	no	yes

The pattern is now visible:

- finite VC dimension explains uniform PAC learnability;
- Occam algorithms explain non-uniform PAC learnability.

Where We Are in Today's Lecture

We have now done the main part of today's plan:

- formalized generalization via expected error and the PAC viewpoint;
- seen the VC-dimension route to *uniform* PAC guarantees;
- seen the Occam route to PAC guarantees without uniformity.

Next:

how do these ideas apply to logical concept classes?

Transition: One More Logical Perspective

At this point, we could return to our running logical examples

$$\mathsf{PL}_{\wedge,\top}[\mathsf{PROP}] \qquad \text{and} \qquad \mathsf{FO}_{\exists,\wedge,\top}[\mathbf{S}]$$

and ask about fitting algorithms that have the PAC property.

Instead, we will do something a bit different.

One More Logical Perspective:
FO-Definable Concept Classes

So Far: Concepts Were Logical Formulas

In Lectures 1 and 2, when we talked about *logical concept classes*, we usually meant:

- the concepts themselves are formulas,
- and examples are structures or assignments satisfying those formulas.

There is another natural perspective:

> *fix one ambient structure A and one first-order formula $\phi(\mathbf{x}, \mathbf{p})$, and let the parameter values $\mathbf{p}$ determine the concepts.*

Definition: FO-Definable Concept Classes over a Fixed Structure

Fix:

- a structure A,
- a first-order formula $\phi(\mathbf{x}, \mathbf{p})$,
  where $\mathbf{x}$ are example variables and $\mathbf{p}$ are parameter variables.

For each parameter tuple $\mathbf{b} \in Dom(A)^{|\mathbf{p}|}$, we obtain a concept

$$c_{(A,\phi,\mathbf{b})}$$

whose positive examples are exactly the tuples $\mathbf{a}$ such that

$$A \models \phi(\mathbf{a}, \mathbf{b}).$$

The resulting concept class is denoted $\mathsf{C}_{(A,\phi)}$.

Main Case Study: the Real Closed Field

Let

$$A = (\mathbb{R}, +, \cdot, <, 0, 1).$$

Then many geometric concept classes can be written as $\mathsf{C}_{(A,\phi)}$:

- **rectangles in the plane:**

$$\phi(x, y; p_1, p_2, p_3, p_4) = p_1 \leq x \wedge x \leq p_2 \wedge p_3 \leq y \wedge y \leq p_4$$

- **disks:**

$$\phi(x, y; p_1, p_2, p_3) = (x - p_1)^2 + (y - p_2)^2 \leq p_3^2$$

Still in the Real Closed Field

The same framework also captures:

- **halfspaces (a.k.a. linear classifiers):**

$$\phi(x_1, \ldots, x_k; p_1, \ldots, p_{k+1}) = p_1 x_1 + \cdots + p_k x_k > p_{k+1}$$

- and many other natural geometric concept classes

Revisiting Our Questions

This fixed-structure perspective lets us revisit all the main questions from the course.

- **Unique characterization:** not much of a story here. Even over $(\mathbb{R}, +, \cdot, <, 0, 1)$, very simple classes such as thresholds can fail to admit finite unique characterizations.
- **Fitting:** here there is a nice complexity-theoretic story.
- **Generalization:** here there is a nice model-theoretic story.

So on the remaining slides, we will focus on fitting and on generalization.

Revisiting Fitting

Fix A and $\phi(\mathbf{x}, \mathbf{p})$, and let E be a labeled sample.

Then fitting means:

> *find a parameter tuple* $\mathbf{b}$ *such that* $A \models \phi(\mathbf{a}, \mathbf{b})$ *for all positive examples* $\mathbf{a}$*, and* $A \not\models \phi(\mathbf{a}, \mathbf{b})$ *for all negative examples.*

Equivalently, Fitting-Existence asks whether the following existential sentence is true in A:

$$\exists \mathbf{p} \Big(\bigwedge_{(\mathbf{a},+) \in E} \phi(\mathbf{a}, \mathbf{p}) \ \wedge \bigwedge_{(\mathbf{a},-) \in E} \neg \phi(\mathbf{a}, \mathbf{p}) \Big).$$

Why the Real Closed Field Is Algorithmically Tame

The key model-theoretic fact is:

Quantifier elimination

Every first-order formula over $(\mathbb{R}, +, \cdot, <, 0, 1)$ is equivalent, over the real numbers, to a quantifier-free formula.

Concretely, this means:

- quantifiers can be eliminated;
- what remains is a Boolean combination of polynomial equalities and inequalities;
- after plugging in the sample points, fitting becomes evaluating an existential sentence over $(\mathbb{R}, +, \cdot, <, 0, 1)$.

What This Means for Our Fixed-Formula Setting

In the textbook's setup, the formula $\phi(\mathbf{x}, \mathbf{p})$ is fixed in advance.

Then for $A = (\mathbb{R}, +, \cdot, <, 0, 1)$:

- Fitting-Existence becomes an existential sentence (hence it is in $\exists\mathbb{R}$).
- In fact, with a *bounded number of variables*, therefore it can be decided in polynomial time (for rational inputs under the usual computation models).

Consequence

Fitting-Existence is in PTIME for first-order concept classes over $(\mathbb{R}, +, \cdot, <, 0, 1)$

Fitting-Construction is more delicate: even when the input examples are rational, a fitting concept may require algebraic parameter values.

Revisiting Generalization: NIP

We now ask the Lecture 3 question again:

> *when do FO-definable concept classes over a fixed structure have finite VC dimension?*

In model theory, a structure A has the *No Independence Property* (NIP) if

$$\mathsf{C}_{(A,\phi)}$$

has finite VC dimension for *every* first-order formula $\phi(\mathbf{x}, \mathbf{p})$.

So NIP is exactly the right ambient-structure condition for uniform VC control in this setting.

Why the Real Closed Field Is Well Behaved

The structure

$$(\mathbb{R}, +, \cdot, <, 0, 1)$$

has NIP. In fact, more generally, all o-minimal structures have NIP.

Therefore, for every first-order formula $\phi(\mathbf{x}, \mathbf{p})$, the concept class $\mathsf{C}_{((\mathbb{R},+,\cdot,<,0,1),\phi)}$ has finite VC dimension.

Learning-theoretic consequence

Every fitting algorithm for such a concept class has the *uniform PAC property*.

This gives one conceptual explanation for why geometric concept classes such as rectangles, halfspaces, and disks generalize well.

Presburger Arithmetic Is Similarly Well Behaved

Another tame setting is Presburger arithmetic:

$$(\mathbb{N}, +, <, 0, 1).$$

It behaves in a very similar spirit:

- it admits quantifier elimination (in a suitable expanded language);
- it has NIP;
- therefore, fixed-formula fitting is again well behaved;
- and FO-definable concept classes over it again have finite VC dimension.

So the real closed field is not an isolated miracle.

A Sharp Contrast: Multiplication on the Natural Numbers

Now let

$$A = (\mathbb{N}, \cdot)$$

and consider the formula

$$\phi(x, p) := \exists y \, (x \cdot y = p).$$

Then $\mathsf{C}_{(A,\phi)}$ has *infinite* VC dimension (i.e., can shatter arbitrarily large sets).

A Sharp Contrast: Multiplication on the Natural Numbers

Now let

$$A = (\mathbb{N}, \cdot)$$

and consider the formula

$$\phi(x, p) := \exists y \, (x \cdot y = p).$$

Then $\mathsf{C}_{(A,\phi)}$ has *infinite* VC dimension (i.e., can shatter arbitrarily large sets).

Why?

Take any finite set of prime numbers $X = \{q_1, \ldots, q_n\}$. For each subset $Y \subseteq X$, let

$$p = \prod_{q \in Y} q.$$

Then $A \models \phi(q, p)$ iff $q \in Y$.

So $(\mathbb{N}, \cdot)$ Does Not Have NIP

The previous slide shows that $(\mathbb{N}, \cdot)$ is *not* NIP.

Hence the FO-definable concept classes over a fixed structure can behave very differently:

- over $(\mathbb{R}, +, \cdot, <, 0, 1)$, they are uniformly VC-bounded and uniformly PAC;
- over $(\mathbb{N}, \cdot)$, even very simple-looking formulas can give infinite VC dimension.

So the ambient structure matters just as much as the formula.

Takeaway

This lecture had three main themes:

- **Formalizing generalization:** expected error, the PAC property, and the uniform PAC property.
- **Two routes to generalization:** finite VC dimension gives uniform PAC guarantees, while Occam algorithms give PAC guarantees without uniformity.
- **Another meeting point of logic and learning:** concept classes of the form $\mathsf{C}_{(A,\phi)}$ over structures such as $(\mathbb{R}, +, \cdot, <, 0, 1)$ or Presburger arithmetic.

In tame structures, fitting and generalization behave well. In structures such as $(\mathbb{N}, \cdot)$, VC dimension can become infinite.

Overview

- **Lecture 1:** from concepts to examples
  using, among other things, some finite model theory.
- **Lecture 2:** from examples to concepts
  with the focus mostly on algorithms and complexity.
- **Lecture 3:** fitting algorithms that provably generalize from the input data
  using probability theory and combinatorics.
- **Lecture 4:** Graph neural networks and their connections to logics with counting
- **Lecture 5:** Interactive models of learning (guessing games)