Logic, Data Examples and Learning

Lecture 5/5: Interactive Learning

Balder ten Cate

Tsinghua University, May 23, 2026

How This Lecture Fits into the Series

In the earlier lectures, we focused on:

- how examples can characterize concepts;
- how to construct fitting concepts from examples;
- and when such concepts generalize.

Today we change the learning model itself:

instead of passively receiving examples, the learner can actively ask questions.

Plan for Today

1. Interactive exact learning (Dana Angluin, 1987).
2. Warmups: $\mathrm{PL}_{\wedge,\top}$ and MonDNF formulas.
3. An exact learning algorithm for $\mathrm{FO}_{\exists,\wedge,\top}$-sentences.
4. Wrap up

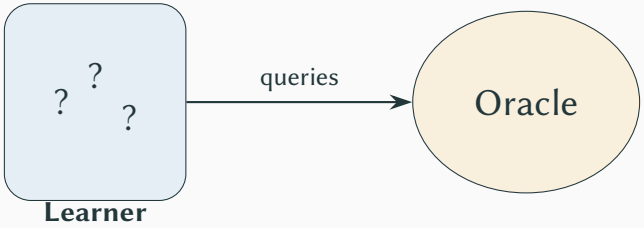
A main theme is:
interaction can make learning easier.

Interactive Learning

Motivation

In interactive learning, the learner is given more flexibility:

- instead of living with the examples it happens to receive,
- it can actively request suitable information.



One may imagine the oracle to be a domain expert.

Historic note. The classic exact-learning framework is due to Dana Angluin (1987).

Oracles

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

We again assume there is a unique target concept $c^* \in C$ to be learned.

We consider two types of oracle queries:

Membership Oracle

input: an unlabeled example $e \in \mathcal{X}$

output: the label of e according to c^*, that is, whether $e \in \lambda(c^*)$

Equivalence Oracle

input: a hypothesis $c_h \in C$

output: *yes* if $\lambda(c_h) = \lambda(c^*)$, and otherwise *no* plus a counterexample

$$e \in \lambda(c_h) \oplus \lambda(c^*). \qquad (\oplus = \text{symmetric difference})$$

Exact Learning

Definition (Exact Learning Algorithm)

Let $\mathsf{C} = (C, \mathcal{X}, \lambda)$ be a concept class. An exact learning algorithm for C is an algorithm that has access to

- a membership oracle, and
- an equivalence oracle.

It must terminate after finitely many steps and output a concept $c \in C$ that is equivalent to the target concept c^*.

Notes:

- No initial sample is provided; the learner requests everything it needs.
- One can also study models with only membership queries or only equivalence queries.

Polynomial-Time Exact Learning

Definition (Polynomial-Time Exact Learning)

An exact learning algorithm runs in polynomial time if there exists a polynomial p such that, at every point during the run, the time spent so far is bounded by

$$p\big(|c^*|, s\big),$$

where s is the size of the largest counterexample returned so far.

Why do we mention the size of counterexamples explicitly?

- Otherwise the oracle could force the learner to process huge objects.
- It is not enough to bound only the final total running time by $p(|c^*|, s)$: the learner could first do brute force search and only later force the oracle to output a gigantic counterexample.

Counterexamples

There are two kinds of counterexamples from the equivalence oracle:

Positive Counterexample

$$e \in \lambda(c^*) \setminus \lambda(c_h)$$

The example belongs to the target concept, but not to the current hypothesis.

Negative Counterexample

$$e \in \lambda(c_h) \setminus \lambda(c^*)$$

The example belongs to the hypothesis, but not to the target concept.

So here, the polarity refers to the *target concept*, not to the current hypothesis.

Warmup: $\mathrm{PL}_{\wedge,\top}$

$\mathrm{PL}_{\wedge,\top}$ Formulas

We start with $\mathrm{PL}_{\wedge,\top}$, i.e., conjunctions such as

$$x_1 \wedge x_3 \wedge x_5 \wedge x_6.$$

Notes:

- Every truth assignment t can be viewed as a conjunction φ_t.
- The target formula φ^* is itself just a set of variables that must be true.
- So, at least in principle, learning means figuring out exactly which variables belong to that set.

We will look at two exact-learning algorithms:

- first one using only membership queries;
- then one using only equivalence queries.

Learning $\mathrm{PL}_{\wedge,\top}$ with Membership Queries Only

φ_h := $\top$
for each variable x_i do
    let $t^{(i)}$ be the valuation with
        $t^{(i)}(x_i) = 0$
        $t^{(i)}(x_j) = 1$ for all $j \neq i$
    call memb($t^{(i)}$)
    if the answer is "no" then
        set $\varphi_h := \varphi_h \wedge x_i$
output φ_h

In other words:

φ_h is the conjunction of all x_i for which $t^{(i)} = \underbrace{\langle 1 \dots 101 \dots 1 \rangle}_{i\text{-th position } 0}$ is a negative example.

Why Membership Queries Already Suffice

For each variable x_i, the query assignment $t^{(i)}$ tests exactly whether x_i is required by the target conjunction.

Indeed:

- if x_i occurs in φ^*, then $t^{(i)} \not\models \varphi^*$, so the membership oracle answers *no*;
- if x_i does not occur in φ^*, then all conjuncts of φ^* are still true in $t^{(i)}$, so the oracle answers *yes*.

So after one membership query per variable, the learner knows exactly which variables belong to the target.

Learning $\mathrm{PL}_{\wedge,\top}$ with Equivalence Queries Only

Invariant

We maintain a hypothesis φ_h such that

$$\varphi_h \models \varphi^*.$$

So every counterexample returned to $\mathrm{equiv}(\varphi_h)$ must be positive.

That is the idea:

- we start with the strongest hypothesis;
- every equivalence counterexample tells us how to weaken it;
- and we never have to worry about negative counterexamples at all.

The Equivalence-Query Algorithm for $\mathrm{PL}_{\wedge,\top}$

```
φ_h := conjunction of all variables
repeat forever
    call equiv(φ_h)
    if the answer is "yes" then
        output φ_h and stop
    else
        let t be the counterexample returned by the oracle
        remove from φ_h all variables x with t(x) = 0
```

The update step is exactly:

> *merge the current hypothesis with newly obtained information by taking the GLB (greatest lower bound) with respect to* $\leq_b$.

A Tiny Example Run

Learner: Is $\varphi^* \equiv x_1 \wedge x_2 \wedge x_3 \wedge x_4$?

Oracle: no, and here is a positive counterexample t_{1110}.

Learner updates by removing x_4 and asks whether $\varphi^* \equiv x_1 \wedge x_2 \wedge x_3$.

Oracle: still no, with positive counterexample t_{1011}.

After removing x_2, the learner reaches $x_1 \wedge x_3$, which is the target.

Why the Algorithm Works

If the algorithm terminates, it clearly outputs a conjunction equivalent to φ^*.

Why does it terminate in polynomial time?

- Every counterexample is positive.
- So in every round, at least one variable switches from 1 to 0 in the hypothesis.
- This can happen at most once per variable.

In particular:

$\mathrm{PL}_{\wedge,\top}$ *is exactly learnable in polynomial time with membership queries only.*
$\mathrm{PL}_{\wedge,\top}$ *is exactly learnable in polynomial time with equivalence queries only.*

(Here, by polynomial time, we mean polynomial in $n = |\mathrm{PROP}|$.)

Warmup: MonDNF

A monotone DNF formula is a disjunction of monotone conjunctions, for example

$$(x_1 \wedge x_2) \ \vee \ (x_1 \wedge x_3) \ \vee \ (x_2 \wedge x_4).$$

Notes:

- $\bot$ corresponds to the empty disjunction.
- Again, every truth assignment t can be viewed as a conjunction φ_t.
- As before, the learner maintains the invariant

  $$\varphi_h \models \varphi^*,$$

  so all counterexamples are positive.

Learning MonDNF Formulas

```
φ_h := ⊥
repeat forever
    call equiv(φ_h)
    if the answer is "yes" then
        output φ_h and stop
    else
        let t be the counterexample returned by the oracle
        find a minimal t' ≤_b t with t' ⊨ φ*
        set φ_h = φ_h ∨ φ_t'
```

The minimization of t is done with membership queries: try switching 1s to 0s one by one.

A Tiny Example Run

Learner starts with $\bot$ and asks whether φ^* is empty.

Oracle: no, here is a positive example t_{111}.

Learner minimizes t_{111} with membership queries and obtains t'_{101}.

So it adds the disjunct $x_1 \wedge x_3$ to the hypothesis.

Repeating this process adds one genuine target disjunct at a time until the full MonDNF is learned.

Why the Algorithm Terminates

Two facts drive the proof.

Fact 1

Every disjunct $\varphi_{t'}$ added to the hypothesis must already be a disjunct of the target formula φ^*.

Reason: if $t' \models \varphi^*$, then some target disjunct is satisfied by t', and minimality forces that disjunct to contain exactly the variables of t'.

Fact 2

The learner never adds the same disjunct twice.

So after at most $|\varphi^*|$ rounds, all target disjuncts have been recovered.

Where We Are So Far

Two warmups already show that the exact-learning model is quite rich.

$\mathrm{PL}_{\wedge,\top}$

$\mathrm{PL}_{\wedge,\top}$ is efficiently exactly learnable

- using only membership queries, and also
- using only equivalence queries.

MonDNF

For MonDNF, the standard polynomial-time exact learner uses both types of queries.

- equivalence queries provide positive counterexamples;
- membership queries are then used to minimize them into target disjuncts.

Both are necessary (Abasi et al. 2014; Bshouty–Eiron 2002).

More About Exact Learning

This exact-learning model was introduced by Dana Angluin in 1987.

Membership and equivalence queries often support clever efficient learning algorithms for concept classes that are not learnable in other frameworks (e.g., PAC).

Classic examples (from Angluin's work) include:

- monotone DNF formulas;
- propositional Horn theories;
- regular languages, represented by deterministic finite automata.

Efficient exact learning with membership queries $\Rightarrow$ existence of small uniquely characterizing sets of examples.

Efficient exact learning with equivalence queries $\Rightarrow$ learnability in the "fitting algorithms with the PAC property" sense.

Learning $\mathrm{FO}_{\exists,\wedge,\top}$-Sentences

Main Result

Theorem

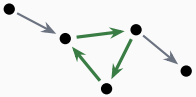
The class of $\mathrm{FO}_{\exists,\wedge,\top}$-sentences is exactly learnable in polynomial time.

Recall:

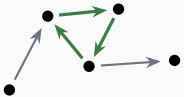
- the class is monotone with respect to the homomorphism pre-order $\leq_{\mathrm{hom}}$;
- canonical examples and canonical concepts exist;
- greatest lower bounds exist in $\leq_{\mathrm{hom}}$;
- those greatest lower bounds are given by direct products.

So structurally, this case looks a lot like $\mathsf{PL}_{\wedge,\top}[\mathsf{PROP}]$ — but the examples are now finite structures.

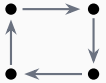
Recap: Fitting Example



positive



positive



negative

$$C_3 \leq_{\text{hom}} B_1, \qquad C_3 \leq_{\text{hom}} B_2, \qquad C_3 \not\leq_{\text{hom}} N.$$

Therefore the canonical concept of C_3 fits this sample.

Recap: Recasting the Fitting Problem

For $\mathsf{FO}_{\exists,\wedge,\top}[\mathbf{S}]$, fitting can be rephrased entirely on the example side.

Goal

Find a finite structure A such that:

- $A \leq_{\hom} B$ for every positive example B;
- $A \not\leq_{\hom} N$ for every negative example N.

Then the canonical concept of A is fitting.

Conversely, every fitting formula gives such a structure via its canonical example.

Recap: Greatest Lower Bounds = Direct Products

For structures A and B over the same schema, the direct product $A \times B$ has:

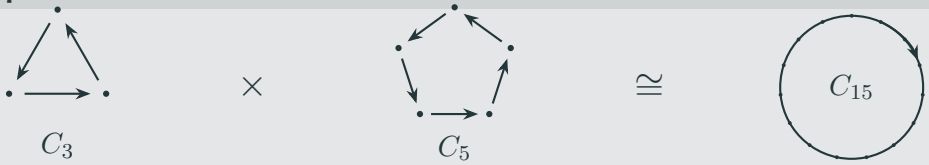
- domain $\mathrm{Dom}(A) \times \mathrm{Dom}(B)$;
- a fact $R((a_1, b_1), \ldots, (a_k, b_k))$ exactly when both

$$R(a_1, \ldots, a_k) \in A \quad \text{and} \quad R(b_1, \ldots, b_k) \in B.$$

Key fact

$A \times B$ is a greatest lower bound of A and B under $\leq_{\mathrm{hom}}$.

Example from Lecture 2



Recap: Exponential Blow Up

- Size multiplies:

$$|A_1 \times \cdots \times A_n| = \prod_i |A_i|.$$

- The Lecture 2 fitting algorithm was exponential: it took the product of all positives at once.

Learning $\mathrm{FO}_{\exists,\wedge,\top}$-Sentences: Main Idea

Every $\mathrm{FO}_{\exists,\wedge,\top}$-sentence can be represented by a finite canonical structure. So instead of thinking syntactically about formulas, we can think structurally about finite models and homomorphisms.

Again, we maintain a hypothesis φ_h such that

$$\varphi_h \models \varphi_t,$$

so every counterexample returned by the equivalence oracle is positive.

The main new ingredient is the merge step:

- for formulas in $\mathrm{PL}_{\wedge,\top}$, we merged using the GLB with respect to $\leq_b$;
- here we merge using the GLB with respect to $\leq_{\mathrm{hom}}$;
- GLB is implemented by the direct product.
- We have to do some more work to stay in polynomial time.

Learning $\mathrm{FO}_{\exists,\wedge,\top}$-Sentences

```
φ_h := the canonical FO_{∃,∧,⊤}-sentence of the 1-point all-facts structure
repeat forever
    call equiv(φ_h)
    if the answer is "yes" then
        output φ_h and stop
    else
        let A be the counterexample structure returned by the oracle
        set P = A_{φ_h} × A                                  direct product = GLB
        find a minimal substructure P' ⊆ P with P' ⊨ φ_t
        set φ_h = φ_{P'}
```

The minimization step uses membership queries.

Why the Product Step Makes Sense

Let A_{φ_h} and A_{φ_t} be the canonical structures of the current hypothesis and the target sentence.

Since $\varphi_h \models \varphi_t$, the target structure maps homomorphically to the hypothesis structure:

$$A_{\varphi_t} \leq_{\text{hom}} A_{\varphi_h}.$$

If the oracle returns a positive counterexample structure A, then also

$$A_{\varphi_t} \leq_{\text{hom}} A.$$

Therefore A_{φ_t} maps to the direct product

$$A_{\varphi_h} \times A,$$

so the canonical sentence of that product is still contained in the target.

Intuitively, the product combines the old and new information.

Two Structural Lemmas

Let

$$\varphi_h^{(0)}, \varphi_h^{(1)}, \varphi_h^{(2)}, \dots$$

be the hypotheses produced by the algorithm, and let $A_h^{(i)} := A_{\varphi_h^{(i)}}$ be their canonical structures. It follows from the construction that

$$A_{\phi_t} \;\leq_{\text{hom}}\; \cdots \;\leq_{\text{hom}}\; A_h^{(2)} \leq_{\text{hom}} A_h^{(1)} \leq_{\text{hom}} A_h^{(0)}$$

The proof of polynomial-time termination rests on two facts:

Lemma 1

Each $A_h^{(i)}$ has at most as many elements as the canonical target structure A_{φ_t}.

Lemma 2

For every $i \geq 0$, the next hypothesis structure $A_h^{(i+1)}$ has strictly more elements than $A_h^{(i)}$.

Proof Idea for Lemma 1

Lemma 1

Each $A_h^{(i)}$ has at most as many elements as the canonical target structure A_{φ_t}.

We know that

$$A_{\phi_t} \leq_{\text{hom}} \cdots \leq_{\text{hom}} A_h^{(2)} \leq_{\text{hom}} A_h^{(1)} \leq_{\text{hom}} A_h^{(0)}$$

Claim: each homomorphism $A_{\varphi_t} \to A_h^{(i)}$ must it be surjective.

- If some element of $A_h^{(i)}$ were not hit, we could delete it.
- The resulting smaller structure would still define a sentence contained in φ_t.
- That would contradict the minimality of the structure chosen in the previous round.

Therefore the number of elements of $A_h^{(i)}$ cannot exceed that of A_{φ_t}.

Proof Idea for Lemma 2

Lemma 2

$A_h^{(i+1)}$ has strictly more elements than $A_h^{(i)}$.

Recall again

$$A_{\phi_t} \leq_{\text{hom}} \cdots \leq_{\text{hom}} A_h^{(2)} \leq_{\text{hom}} A_h^{(1)} \leq_{\text{hom}} A_h^{(0)}$$

In fact, each $A_h^{(i+1)}$ is homomorphically *strictly* below $A_h^{(i)}$.

The argument has two parts.

1. First, the same minimization argument as on the previous slide implies that $A_h^{(i+1)}$ cannot have *fewer* elements than $A_h^{(i)}$.
2. Second, suppose, for a contradiction, that $A_h^{(i+1)}$ has the same domain size as $A_h^{(i)}$. Then (again by minimality of $A_h^{(i)}$) the homomorphism from $A_h^{(i+1)} \to A_h^{(i)}$ is an isomorphism, contradicting $A_h^{(i+1)} <_{\text{hom}} A_h^{(i)}$.

Why This Gives Polynomial-Time Termination

Combine the two lemmas:

- the sequence of canonical hypothesis structures strictly increases in size;
- but no hypothesis structure can be larger than the canonical target structure.

So only polynomially many rounds are possible.

The key moral is:

> *for $\mathrm{FO}_{\exists,\wedge,\top}$-sentences, direct product gives the right merge operation, and minimization is what makes the exact-learning algorithm efficient.*

Only Equivalence Queries?

Can We Drop Membership Queries?

For $\mathrm{PL}_{\wedge,\top}$, the learning algorithm did not use membership queries at all.

Can the same happen for $\mathrm{FO}_{\exists,\wedge,\top}$-sentences?

A clean way to answer this is via PAC learning.

Fact

If a concept class is exactly learnable in polynomial time using only equivalence queries, then it is PAC learnable in polynomial time.

So, from the earlier lectures:

- since $\mathrm{FO}_{\exists,\wedge,\top}$-sentences are not PAC learnable in polynomial time,
- they cannot be exactly learnable in polynomial time with only equivalence queries.

PAC Learning via Oracles

To prove the lemma, it is convenient to reformulate PAC learning using an oracle.

Definition (PAC Oracle Model)

The learner receives as input only $\delta, \varepsilon \in (0, 1)$ and has access to an oracle that, in unit time, returns a labeled example drawn from the distribution $\mathbb{P}$.

We then require that, with probability at least $1 - \delta$, the learner outputs a concept c whose error with respect to c^* and $\mathbb{P}$ is at most ε.

This oracle-based version is equivalent to the "fitting algorithm with PAC property" version of learning from Lecture 3.

It is known that $\mathsf{FO}_{\exists,\wedge,\top}$-sentences are not efficiently PAC learnable.

Why Exact Learning with Equivalence Queries Gives PAC Learning

Suppose we have an exact learner A that uses only equivalence queries.

We simulate each equivalence query by drawing enough random examples:

- when A proposes a hypothesis c in round i,
- draw
$$n_i = \left\lceil \frac{1}{\varepsilon}\left(\ln(1/\delta) + i \ln 2\right) \right\rceil$$
random labeled examples;
- if one of them refutes c, answer *no* and feed that example back as a counterexample;
- if none refute c, answer *yes*.

Intuitively, a genuinely bad hypothesis is very unlikely to survive this test.

The Probability Calculation

Suppose a hypothesis c in round i has true error greater than ε.

Then the probability that all n_i random examples are still consistent with c is at most

$$(1-\varepsilon)^{n_i} \leq e^{-\varepsilon n_i}.$$

Therefore, the probability of *ever* accepting such a bad hypothesis is bounded by

$$\sum_{i \geq 1} e^{-\varepsilon n_i} \leq \sum_{i \geq 1} \frac{\delta}{2^i} \leq \delta.$$

So with probability at least $1-\delta$, the accepted hypothesis has error at most ε.

What This Implies for $\mathrm{FO}_{\exists,\wedge,\top}$-Sentences

We obtain two complementary conclusions.

Negative

$\mathrm{FO}_{\exists,\wedge,\top}$-sentences are *not* exactly learnable in polynomial time with only equivalence queries.

Positive: PAC learnability with membership queries

If membership queries are available in addition to random examples, then $\mathrm{FO}_{\exists,\wedge,\top}$-sentences become PAC learnable in polynomial time.

some details hidden under the carpet

(cf. tC, Funk, Jung, Lutz, *On the Non-Efficient PAC Learnability of Conjunctive Queries*, **IPL 2024**).

Five Lectures in One Picture

- **Lecture 1: from concepts to examples.** We asked when a logical concept can be uniquely characterized by finitely many examples.
- **Lecture 2: from examples to concepts.** We studied fitting algorithms, extremal fitting, and the complexity of finding succinct fittings.
- **Lecture 3: from fitting to generalization.** We formalized PAC learning, and saw two roads to guarantees: VC dimension and Occam bounds.
- **Lecture 4: logic and graph learning.** We looked at GNNs through the lens of modal, arithmetic, and first-order counting logics.
- **Lecture 5: interactive learning.** We saw how membership and equivalence queries can dramatically change what is learnable.

Running examples: $\mathsf{PL}_{\wedge,\top}$, $\mathsf{PL}_{\wedge,\vee,\top,\bot}$, $\mathsf{FO}_{\exists,\wedge,\top}$.

Same methodology applies to (fragments of) many other languages.

Recent Work: Database Queries

- **Unique characterizability / exact learning.**
  ten Cate and Dalmau, *Conjunctive Queries: Unique Characterizations and Exact Learnability*, **TODS 2022**.
- **Extremal fitting.**
  ten Cate, Dalmau, Funk, and Lutz, *Extremal Fitting Problems for Conjunctive Queries*, **PODS 2023**. Best Paper
- **PAC learnability.**
  ten Cate, Funk, Jung, and Lutz, *On the Non-Efficient PAC Learnability of Conjunctive Queries*, **IPL 2024**.
- **Repairs / interactive query refinement.**
  ten Cate, Kolaitis, and Lutz, *Query Repairs*, **ICDT 2025**.

Historic note. Example-driven querying has a long history in databases (*Query by Example* paradigm, Zloof 1970s).

Recent Work: Modal and Description Logics

- **Modal unique characterizations.**
  tC and Koudijs, *Characterising Modal Formulas with Examples*, **TOCL 2024**.
- **Exact learning under ontologies.**
  Funk, Jung, and Lutz, *Frontiers and Exact Learning of ELI Queries under DL-Lite Ontologies*, **IJCAI 2022**.
- **PAC-style learning for DL concepts.**
  tC, Funk, Jung, and Lutz, *SAT-Based PAC Learning of Description Logic Concepts*, **IJCAI 2023**. Distinguished Paper.
- **Power and limits of examples.**
  tC, Koudijs, and Ozaki, *On the Power and Limitations of Examples for Description Logic Concepts*, **IJCAI 2024**.

Historic note. There is also a longer background literature on concept learning in description logics, for example Lehmann and Hitzler, *Machine Learning* 2010.

Recent Work: Temporal Logic

- **Learning from example traces.**
  Neider and Gavran, *Learning Linear Temporal Properties*, **FMCAD 2018**.
  Fijalkow and Lagarde, *The Complexity of Learning Linear Temporal Formulas from Examples*, **PMLR 2021**.
  Bordais, Neider, and Roy, *The Complexity of Learning LTL, CTL and ATL Formulas*, **STACS 2025**.
- **Generating examples for LTL formulas.**
  tC, Fisman, Ohayon, and Sestic, *Characterizing LTL Formulas by Examples*, submission.

Here the choice of *what counts as an example* becomes interesting: finite traces, infinite traces, transfinite traces, schematic examples.