Week 4: Bias, Complexity, and Computation

Non-uniform learning, efficient PAC learning, and hardness

Tianhao Wang
tianhaowang@ucsd.edu

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

Contents

Bridge from Week 3

Where we left off

Week 3. The batch/PAC setting for binary classification.

Instance space, labels, hypothesis class:

$$\mathcal{X} \text{ instance space}, \qquad \mathcal{Y} = \{0, 1\}, \qquad \mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}.$$

Unknown source distribution and i.i.d. sample:

$$\mathcal{D} \in \Delta(\mathcal{X} \times \mathcal{Y}), \qquad S = ((x_1, y_1), ..., (x_n, y_n)) \sim \mathcal{D}^n.$$

Risk and empirical risk:

$$L_{\mathcal{D}}(h) = \Pr_{(x,y)\sim\mathcal{D}}[h(x) \neq y], \qquad L_S(h) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[h(x_i) \neq y_i].$$

A learning rule is a map $A : S \mapsto A(S)$, usually outputting a predictor $A(S) : \mathcal{X} \to \mathcal{Y}$.

Fundamental Theorem: one-slide recap

Fundamental Theorem of PAC Learning

For binary classification,

$$\mathrm{VCdim}(\mathcal{H}) < \infty \quad \Leftrightarrow \quad \mathcal{H} \quad \text{is PAC learnable.}$$

Moreover, in the agnostic setting, for every $\epsilon, \delta \in (0,1)$, if $n = O\left(\frac{\mathrm{VCdim}(\mathcal{H}) + \log(1/\delta)}{\epsilon^2}\right)$, then with probability at least $1-\delta$ over $S \sim \mathcal{D}^n$,

$$L_{\mathcal{D}}(\mathsf{ERM}_{\mathcal{H}}(S)) \leq \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon.$$

In the realizable setting, the dependence on ϵ improves to roughly $1/\epsilon$.

- Forward direction ($\mathrm{VCdim}(\mathcal{H}) < \infty \Rightarrow \mathcal{H}$ PAC learnable): proved last week.
- Converse ($\mathrm{VCdim}(\mathcal{H}) = \infty \Rightarrow \mathcal{H}$ not PAC learnable): stated, not proved.
- NFL is the lower-bound tool (HW 3, Problem B).

No-Free-Lunch (HW 3, Problem B)

Let A be any learning rule for binary classification, and let $n < |\mathcal{X}|/2$. Then there exists a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0, 1\}$ such that:

(i) some $f^\star : \mathcal{X} \to \{0, 1\}$ satisfies $L_{\mathcal{D}}(f^\star) = 0$;

(ii) $\Pr_{S \sim \mathcal{D}^n}[L_{\mathcal{D}}(A(S)) \geq 1/8] \geq 1/7$.

Interpretation.

- With $n < |\mathcal{X}|\ /2$ samples, the learner sees fewer than half the points of $\mathcal{X}$.
- On unseen points, the adversary can set $f^\star$ to anything.
- To avoid this, we fix a class $\mathcal{H} \subsetneq \{0, 1\}^{\mathcal{X}}$ and assume $f^\star \in \mathcal{H}$.

Bias-complexity decomposition

Fixing $\mathcal{H}$ is the learner's **inductive bias**, and it has a cost.

Let $L^\star_{\mathcal{D}}$ be the minimum risk achievable by any predictor:

$$L^\star_{\mathcal{D}} = \inf_{f:\mathcal{X}\to\mathcal{Y}} L_{\mathcal{D}}(f).$$

For any $\hat{h} \in \mathcal{H}$,

$$L_{\mathcal{D}}\left(\hat{h}\right) - L^\star_{\mathcal{D}} = \underbrace{\inf_{h\in\mathcal{H}} L_{\mathcal{D}}(h) - L^\star_{\mathcal{D}}}_{\text{approximation / bias}} + \underbrace{L_{\mathcal{D}}\left(\hat{h}\right) - \inf_{h\in\mathcal{H}} L_{\mathcal{D}}(h)}_{\text{estimation / complexity}}.$$

Larger $\mathcal{H}$ shrinks approximation but inflates estimation.

The FT's estimation rate depends on $\mathcal{H}$ as a whole, not on individual hypotheses within it.

Simpler hypotheses should be easier to learn

Intuition. A simpler hypothesis should need fewer samples to learn.

> **Example.**
>
> Inside the class of decision trees of depth ≤ 10, a stump (depth 1) should be learnable from far fewer samples than a full depth-10 tree.

But the FT is uniform. The same sample complexity is required whether the hypothesis we compete against is simple or complex.

Non-uniform learnability. Let the sample complexity depend on h:

$$n_{\mathcal{H}}(\epsilon, \delta, h) \quad \text{instead of} \quad n_{\mathcal{H}}(\epsilon, \delta).$$

Goal. The rate for h should reflect the complexity of h itself, not the richness of $\mathcal{H}$. A simple h is cheap to learn even when $\mathcal{H}$ is large.

Today's three parts

- **Part 1.** Non-uniform learning: rates that favor simpler hypotheses.
- **Part 2.** Computational hardness: even when a class is statistically learnable, ERM may not be tractable.
- **Part 3.** Improper hardness: some classes stay hard even if we allow any predictor.

Non-uniform learning

Starting point: fixed-hypothesis concentration

For a fixed predictor h, define

$$Z_i = \mathbf{1}[h(x_i) \neq y_i].$$

Then

$$L_S(h) = \frac{1}{n}\sum_{i=1}^{n} Z_i, \qquad \mathbb{E}[Z_i] = L_{\mathcal{D}}(h).$$

Hoeffding's inequality gives, for any $t > 0$,

$$\mathbb{P}[L_{\mathcal{D}}(h) > L_S(h) + t] \leq e^{-2nt^2}.$$

Equivalently, with probability at least $1 - \delta_h$,

$$L_{\mathcal{D}}(h) \leq L_S(h) + \sqrt{\frac{\log(1/\delta_h)}{2n}}.$$

The subscript h on δ_h is deliberate: soon each hypothesis gets its own failure probability.

Finite class: flat union bound

Let $\mathcal{H}$ be finite. Assign the same failure probability to each hypothesis:

$$\delta_h = \frac{\delta}{|\mathcal{H}|}.$$

Hoeffding for each h, then union bound: with probability at least $1 - \delta$, for every $h \in \mathcal{H}$,

$$L_{\mathcal{D}}(h) \leq L_S(h) + \sqrt{\frac{\log|\mathcal{H}| + \log(1/\delta)}{2n}}.$$

The penalty $\log|\mathcal{H}|$ is the same for every h. It does not reflect that we wanted simpler h to be cheaper to learn.

Weighted union bound: the idea

Why does flat give $\log|\mathcal{H}|$? Because it splits δ **equally**: $\delta_h = \delta/|\mathcal{H}|$.

Equal splitting uses no information about h. If we prefer **simpler h** a priori, we should allocate **more failure probability** to them.

Encode the preference as weights:

$$p(h) \geq 0, \qquad \sum_{h \in \mathcal{H}} p(h) \leq 1, \qquad \delta_h = p(h)\delta.$$

- **Large $p(h)$**: h is simple or a priori plausible.
- **Small $p(h)$**: h is complex or a priori implausible.

$\sum_h p(h) \leq 1$ preserves the total failure probability δ; only the allocation across h changes.

Larger $p(h)$ gives larger δ_h, and thus smaller penalty for h.

Weighted union bound

Derivation.

- Hoeffding for each h at failure probability $\delta_h = p(h)\delta$:

$$\mathbb{P}\left[L_{\mathcal{D}}(h) > L_S(h) + \sqrt{\log(1/\delta_h)/(2n)}\right] \leq p(h)\delta.$$

- Union bound: $\sum_h p(h)\delta \leq \delta$.
- Unpack: $\log(1/\delta_h) = \log(1/p(h)) + \log(1/\delta)$.

Weighted finite-class bound

With probability at least $1-\delta$ over $S \sim \mathcal{D}^n$, for all $h \in \mathcal{H}$ with $p(h) > 0$,

$$L_{\mathcal{D}}(h) \leq L_S(h) + \sqrt{\frac{\log(1/p(h)) + \log(1/\delta)}{2n}}.$$

Complexity from the weight

Recall the penalty: $\sqrt{\frac{\log(1/p(h))+\log(1/\delta)}{2n}}$.

Define

$$\mathsf{complexity}(h) := \log \frac{1}{p(h)}.$$

- Simple or a priori plausible h: **small** $\mathsf{complexity}(h)$.
- Complex or implausible h: **large** $\mathsf{complexity}(h)$.

Rewrite the bound:

$$L_{\mathcal{D}}(h) \leq L_S(h) + \sqrt{\frac{\mathsf{complexity}(h) + \log(1/\delta)}{2n}}.$$

What the prior means

p encodes our preferences over $\mathcal{H}$; it is **chosen by us, not inferred from data**.

- Chosen **before** seeing the sample.
- Encodes what we consider simple or plausible.
- Larger $p(h)$: stronger commitment that h deserves a tight bound.

The theorem holds for every $\mathcal{D}$ and every valid p.

p is a formal version of inductive bias.

Example: two choices of prior

Cardinality prior: $p(h) = 1/|\mathcal{H}|$.

$$\mathsf{complexity}(h) = \log|\mathcal{H}| \quad \text{for every } h.$$

Recovers the flat finite-class bound.

Description-length prior: $p(h) = 2^{-|d(h)|}$, where $|d(h)|$ is the length of a binary description of h.

$$\mathsf{complexity}(h) = |d(h)| \cdot \log 2.$$

For example, $d(h)$ could be:

- the binary digits of an integer index for h;
- the source code of a program that computes h.

Shorter description, smaller penalty. (Occam's razor.)

Does $2^{-|d(h)|}$ satisfy $\sum_h p(h) \leq 1$? Next slide: Kraft inequality.

Description and prefix-free codes

- **Description.** $d : \mathcal{H} \to \{0,1\}^*$ assigns each h a binary string.
  - e.g., $d(h_1) = 00$, $d(h_2) = 01$, $d(h_3) = 10$, $d(h_4) = 11$.
- **Prefix.** One binary string is a prefix of another if the latter starts with the former.
  - e.g., 01 is a prefix of 0110.
- **Prefix-free description.** No $d(h)$ is a prefix of any other $d(h')$.
  - **Prefix-free:** $a \mapsto 0$, $b \mapsto 10$, $c \mapsto 11$.
  - **Not prefix-free:** $a \mapsto 0$, $b \mapsto 01$ (since 0 is a prefix of 01).

Kraft inequality

Let $d : \mathcal{H} \to \{0,1\}^*$ be a prefix-free description. Then

$$\sum_{h \in \mathcal{H}} 2^{-|d(h)|} \leq 1.$$

Proof. Fix L and draw a uniform random binary string σ of length L. For every h with $|d(h)| \leq L$,

$$\mathbb{P}[\sigma \text{ starts with } d(h)] = 2^{-|d(h)|}.$$

Prefix-free $\Longrightarrow$ these events are pairwise disjoint, so

$$\sum_{h : |d(h)| \leq L} 2^{-|d(h)|} \leq 1.$$

Letting $L \to \infty$ gives the full sum.

$p(h) = 2^{-|d(h)|}$ is a valid prior whenever d is prefix-free.

MDL: Minimum Description Length

Weighted bound:

$$L_{\mathcal{D}}(h) \leq L_S(h) + \sqrt{\frac{\mathsf{complexity}(h) + \log(1/\delta)}{2n}}.$$

- In the realizable setting, some h satisfies $L_S(h) = 0$.
- For such h, the bound depends only on $\mathsf{complexity}(h)$ and is monotone in it.
- With $p(h) = 2^{-|d(h)|}$, $\mathsf{complexity}(h) = |d(h)| \cdot \log 2$.
- The **tightest bound among consistent** h comes from the **shortest** $|d(h)|$.

Rule.

$$\mathsf{MDL}_p(S) = \arg \max_{h: L_S(h)=0} p(h) = \arg \min_{h: L_S(h)=0} |d(h)|.$$

Among all predictors that explain the sample perfectly, choose the one with the shortest description.

Occam's razor as a learning rule.

MDL guarantee

Assume realizability: some $h^\star \in \mathcal{H}$ has $L_{\mathcal{D}}(h^\star) = 0$. Then $L_S(h^\star) = 0$ almost surely.

- $h^\star$ is consistent, so $h^\star$ is a candidate in MDL's argmin.
- $\Longrightarrow \left|d\big(\mathsf{MDL}_p(S)\big)\right| \leq |d(h^\star)|$.

Apply the weighted bound at $\mathsf{MDL}_p(S)$; since $L_S\big(\mathsf{MDL}_p(S)\big) = 0$,

$$L_{\mathcal{D}}\big(\mathsf{MDL}_p(S)\big) \leq \sqrt{\frac{|d(h^\star)| \cdot \log 2 + \log(1/\delta)}{2n}}.$$

MDL's error is controlled by the description length of the best hypothesis.

Universal MDL

Goal. Apply MDL to every computable function (one computed by some algorithm), not a pre-specified class.

Need a description language that is:

- **Universal**: can express any computable function.
- **Prefix-free**: valid programs form a prefix-free code (so Kraft gives a valid prior).

Let U be such a language. Define

$$d_U(h) = \min\{|\sigma| : U(\sigma) = h\}.$$

$d_U(h)$ is the **Kolmogorov complexity** of h: the length of the shortest program in U that outputs h.

Then $p(h) = 2^{-d_U(h)}$ is a valid prior over computable functions (picking one shortest program per h).

MDL can learn every computable target non-uniformly:

$$n_{\mathcal{H}}(\epsilon, \delta, h^\star) = O\left(\frac{d_U(h^\star) + \log(1/\delta)}{\epsilon^2}\right).$$

Why no contradiction with VC theory?

The class of all computable functions has **infinite VC dimension**. (For any finite set, every labeling is computed by a lookup-table program.)

But the Fundamental Theorem is about **uniform** sample complexity:

$$\forall \epsilon, \delta \quad \exists n_{\mathcal{H}}(\epsilon, \delta) \quad \forall h^\star \in \mathcal{H}.$$

MDL gives **non-uniform** sample complexity:

$$\forall \epsilon, \delta, h^\star \quad \exists n_{\mathcal{H}}(\epsilon, \delta, \boldsymbol{h^\star}).$$

Dependence on $h^\star$ avoids the contradiction. Next: formalize the distinction via **quantifier order**.

Uniform vs non-uniform learnability

Formal definitions. The two notions differ in the **order of quantifiers**.

Agnostic PAC learnability

There exists a rule A such that $\forall \epsilon, \delta > 0$, $\exists n_{\mathcal{H}}(\epsilon, \delta)$, $\forall \mathcal{D}$, $\forall h \in \mathcal{H}$:

$$\mathbb{P}_{S \sim \mathcal{D}^n}[L_{\mathcal{D}}(A(S)) \leq L_{\mathcal{D}}(h) + \epsilon] \geq 1 - \delta.$$

Non-uniform learnability

There exists a rule A such that $\forall \epsilon, \delta > 0$, $\forall h \in \mathcal{H}$, $\exists n_{\mathcal{H}}(\epsilon, \delta, h)$, $\forall \mathcal{D}$:

$$\mathbb{P}_{S \sim \mathcal{D}^n}[L_{\mathcal{D}}(A(S)) \leq L_{\mathcal{D}}(h) + \epsilon] \geq 1 - \delta.$$

The learner is the same; the sample size may depend on h.

SRM over hypotheses (countable $\mathcal{H}$)

Structural Risk Minimization (SRM, Vapnik): the general rule for non-uniform learning.

From weighted Hoeffding, the **penalty** (two-sided, at confidence δ):

$$\epsilon(n, \delta, p(h)) := \sqrt{\frac{\mathsf{complexity}(h) + \log(2/\delta)}{2n}}, \quad \mathsf{complexity}(h) = \log(1/p(h)).$$

SRM rule:

$$\hat{h} = \mathsf{SRM}_p(S) = \arg\min_h \{L_S(h) + \epsilon(n, \delta, p(h))\}.$$

Balances fit ($L_S(h)$) against complexity ($\epsilon(n, \delta, p(h))$). No realizability assumption.

MDL is the special case: **realizable setting** + **description-length prior** $p(h) = 2^{-|d(h)|}$.

- Then the argmin degenerates to $\arg\min_{L_S(h)=0} |d(h)|$.

SRM theorem over hypotheses

Hypothesis-level SRM

Let $\mathcal{H}$ be countable and $p : \mathcal{H} \to [0,1]$ satisfy $\sum_{h\in\mathcal{H}} p(h) \le 1$. For any $\mathcal{D}$ and any $\delta \in (0,1)$, with probability at least $1-\delta$ over $S \sim \mathcal{D}^n$, $\hat{h} = \mathsf{SRM}_p(S)$ satisfies

$$L_{\mathcal{D}}\left(\hat{h}\right) \le \inf_{h\in\mathcal{H}} \{L_{\mathcal{D}}(h) + 2\epsilon(n,\delta,p(h))\}.$$

Proof sketch. Two-sided uniform convergence + optimality of $\hat{h}$ for $L_S + \epsilon$.

Interpretation.

- **Oracle bound.** Competes with the best (fit + complexity) tradeoff in $\mathcal{H}$, without needing to know which h achieves it.
- **Automatic model selection.** Each h pays its own penalty $\epsilon(n,\delta,p(h))$; simple h get tighter bounds.
- **Recovers MDL** when $\mathcal{D}$ is realizable and $p(h) = 2^{-|d(h)|}$: the infimum is achieved at the shortest consistent h^*.

Prior over classes

SRM over hypotheses needed $\sum_{h \in \mathcal{H}} p(h) \leq 1$, forcing $\mathcal{H}$ to be **countable**.

But many natural classes are **uncountable**: linear classifiers in $\mathbb{R}^d$, neural nets with a given architecture, polynomials of degree at most r.

Observation. These classes decompose into a countable family of **relatively simple** subclasses indexed by a **complexity level** r:

$$\mathcal{H} = \bigcup_{r=1}^{\infty} \mathcal{H}_r, \quad \mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \cdots$$

"Relatively simple" means each $\mathcal{H}_r$ admits a uniform-convergence bound (e.g., finite VC dimension).

- $\mathcal{H}_r$ = polynomials of degree at most r $\quad (r = \text{degree})$
- $\mathcal{H}_r$ = neural nets with at most r units $\quad (r = \text{width})$
- $\mathcal{H}_r$ = predictors using at most r features $\quad (r = \text{sparsity})$

Idea. Put the prior on the **countable index** r: $p_r \geq 0$, $\sum_r p_r \leq 1$. Apply a VC bound within each $\mathcal{H}_r$, combine via a weighted union bound over r.

SRM over classes (uncountable $\mathcal{H}$)

Prior $p_r \geq 0$ for $r = 1, 2, \ldots$, $\sum_{r=1}^{\infty} p_r \leq 1$. Set $\delta_r = p_r \delta$.

Per-class VC bound + weighted union bound over r give the two-sided uniform convergence: with probability at least $1 - \delta$, for all $r \geq 1$ and all $h \in \mathcal{H}_r$,

$$|L_{\mathcal{D}}(h) - L_S(h)| \leq \epsilon_{\mathrm{VC}}(n, \delta, r) := C\sqrt{\frac{\mathrm{VCdim}(\mathcal{H}_r) + \log(1/p_r) + \log(1/\delta)}{n}}.$$

The penalty combines two sources of complexity:

- $\mathrm{VCdim}(\mathcal{H}_r)$: within-class capacity (difficulty of identifying h inside $\mathcal{H}_r$).
- $\log(1/p_r)$: across-class cost (prior mass allocated to $\mathcal{H}_r$).

SRM **rule:**

$$\hat{h} = \arg \min_{r \geq 1, h \in \mathcal{H}_r} \{L_S(h) + \epsilon_{\mathrm{VC}}(n, \delta, r)\}.$$

Class-level SRM

Let $\mathcal{H} = \bigcup_{r=1}^{\infty} \mathcal{H}_r$ with $\mathrm{VCdim}(\mathcal{H}_r) < \infty$ for all r, and $p_r \geq 0$ with $\sum_{r=1}^{\infty} p_r \leq 1$. For any $\mathcal{D}$ and any $\delta \in (0,1)$, with probability at least $1-\delta$ over $S \sim \mathcal{D}^n$, $\hat{h}$ satisfies

$$L_{\mathcal{D}}\left(\hat{h}\right) \leq \inf_{r \geq 1, h \in \mathcal{H}_r} \left\{ L_{\mathcal{D}}(h) + 2C\sqrt{\frac{\mathrm{VCdim}(\mathcal{H}_r) + \log(1/p_r) + \log(1/\delta)}{n}} \right\}.$$

Proof sketch. Two-sided uniform convergence + optimality of $\hat{h}$ for $L_S + \epsilon_{\mathrm{VC}}$.

Interpretation.

- **Oracle over classes.** Competes with the best (fit + complexity) tradeoff across all r and $h \in \mathcal{H}_r$.
- **Automatic model selection.** Chooses the complexity level r that best balances VC cost against empirical fit.

Example: polynomial degree

Let $\mathcal{H}_r$ be polynomial threshold functions of degree at most r in d variables; $\mathcal{H} = \bigcup_{r=1}^{\infty} \mathcal{H}_r$.

$\mathcal{H}$ has infinite VC dimension (polynomials shatter any finite set), so uniform PAC is impossible.

Degree-r monomials embed $h \in \mathcal{H}_r$ as a halfspace in $\mathbb{R}^{\binom{d+r}{r}}$, so $\mathrm{VCdim}(\mathcal{H}_r) = O\left(\binom{d+r}{r}\right)$.

Choose prior $p_r = 2^{-r}$. The SRM penalty within $\mathcal{H}_r$:

$$\epsilon_{\mathrm{VC}}(n, \delta, r) = O\left(\sqrt{\frac{\binom{d+r}{r} + r + \log(1/\delta)}{n}}\right).$$

VC-dominated: for fixed d and growing r, $\binom{d+r}{r} \approx r^d/d!$ grows much faster than the prior cost r.

For target $h^* \in \mathcal{H}_{r^*}$, sample complexity to reach ϵ error:

$$n(\epsilon, \delta, h^*) = O\left(\frac{\binom{d+r^*}{r^*} + r^* + \log(1/\delta)}{\epsilon^2}\right).$$

Infinite-VC class, yet each target is learnable; sample complexity tracks the target's degree r^*.

Characterization of non-uniform learnability

So far: SRM gives non-uniform learnability whenever $\mathcal{H} = \bigcup_{r=1}^{\infty} \mathcal{H}_r$ with each $\mathcal{H}_r$ finite-VC. Is this structural property **necessary**?

Characterization

A binary class $\mathcal{H}$ is non-uniformly learnable if and only if

$$\mathcal{H} = \bigcup_{r=1}^{\infty} \mathcal{H}_r \quad \text{with} \quad \mathrm{VCdim}(\mathcal{H}_r) < \infty \quad \text{for all } r.$$

- ($\Leftarrow$) Sufficiency: SRM with any summable prior p_r (just proved).
- ($\Rightarrow$) Necessity: if $\mathcal{H}$ is not such a union, a no-free-lunch-style argument defeats every learner.

Compare to the Fundamental Theorem (PAC):

- **PAC:** learnable iff $\mathrm{VCdim}(\mathcal{H}) < \infty$ (a single finite-VC class).
- **Non-uniform:** iff countable union of finite-VC classes. Strictly richer; includes all computable functions.

Characterization: non-uniform iff $\mathcal{H} = \bigcup_{r=1}^{\infty} \mathcal{H}_r$ with $\mathrm{VCdim}(\mathcal{H}_r) < \infty$. SRM realizes this with a **discrete prior**.

Limitation of SRM. The prior is always **discrete**: either $p(h) > 0$ on individual hypotheses, or $p_r > 0$ on subclasses.

What if we want a **continuous prior** over $\mathcal{H}$ (e.g., Gaussian over neural-net weights)?

Idea. Encode inductive bias as a distribution P over hypotheses (possibly continuous).

PAC-Bayes framework. Output a distribution Q over $\mathcal{H}$, not a single h.

- Prior P: fixed before S.
- Posterior Q: chosen after S. Risk: $L_{\mathcal{D}}(Q) = \mathbb{E}_{h \sim Q}[L_{\mathcal{D}}(h)]$.

KL divergence as complexity

Kullback-Leibler divergence between posterior Q and prior P:

$$\mathrm{KL}(Q \parallel P) = \mathbb{E}_{h \sim Q} \log \frac{q(h)}{p(h)}.$$

(Use densities q, p when P, Q are continuous; probabilities otherwise.)

Properties.

- $\mathrm{KL}(Q \parallel P) \geq 0$, with equality iff $Q = P$.
- Asymmetric: $\mathrm{KL}(Q \parallel P) \neq \mathrm{KL}(P \parallel Q)$ in general.
- $\mathrm{KL}(Q \parallel P) = \infty$ if Q puts mass where P does not.

Role in PAC-Bayes. $\mathrm{KL}(Q \parallel P)$ measures how far the posterior moves from the prior; this is the **complexity penalty**, replacing $\log(1/p(h))$ from SRM/MDL.

Sanity check. Discrete P and $Q = \delta_h$: $\mathrm{KL}(\delta_h \parallel P) = \log(1/P(h))$. Recovers the SRM/MDL penalty.

PAC-Bayes theorem

PAC-Bayes

For any prior P, with probability at least $1 - \delta$ over $S \sim \mathcal{D}^n$, simultaneously for all posteriors Q,

$$L_{\mathcal{D}}(Q) \leq L_S(Q) + \sqrt{\frac{\mathrm{KL}(Q \parallel P) + \log(2n/\delta)}{2(n-1)}}.$$

- **Data-dependent posterior.** Q may be chosen after observing S; the KL term is the price for this flexibility.
- **Constrained by P's support.** If $P(h) = 0$ but Q puts mass at h, then $\mathrm{KL}(Q \parallel P) = \infty$ and the bound is vacuous. Only hypotheses in the support of P are available to the learner.
- **Fit-complexity tradeoff.** Concentrating Q on data-fitting h lowers $L_S(Q)$ but pulls away from P, raising KL.
- **Simultaneous over Q.** One high-probability event controls every posterior at once.

PAC-Bayes learning rule

Natural rule: minimize the RHS of the bound over Q:

$$\hat{Q} = \arg\min_{Q} \left\{ L_S(Q) + \sqrt{\frac{\mathrm{KL}(Q \,\|\, P) + \log(2n/\delta)}{2(n-1)}} \right\}.$$

Relaxed form with tunable $\lambda > 0$:

$$\hat{Q}_\lambda = \arg\min_{Q} \{ L_S(Q) + \lambda \cdot \mathrm{KL}(Q \,\|\, P) \}.$$

Closed-form solution: Gibbs / Boltzmann posterior.

$$\hat{Q}_\lambda(h) \propto P(h) \exp(-L_S(h)/\lambda).$$

The prior P is reweighted by data fit, with temperature λ.

- $\lambda \to 0$: $\hat{Q}$ concentrates on ERM (pure fit).
- $\lambda \to \infty$: $\hat{Q} = P$ (pure prior, no fit).
- Intermediate λ: mixture of inductive bias and data fit.

PAC-Bayes recovers earlier bounds

Finite class, uniform prior $P(h) = 1/|\mathcal{H}|$. For $Q = \delta_h$ (point mass on h),

$$\mathrm{KL}(Q \,\|\, P) = \log|\mathcal{H}|.$$

PAC-Bayes recovers a finite-class penalty.

Non-uniform discrete prior. For $Q = \delta_h$,

$$\mathrm{KL}(\delta_h \,\|\, P) = -\log p(h),$$

recovering an MDL/SRM-style penalty.

> **PAC-Bayes vs Bayesian inference**
>
> Both produce Gibbs-looking distributions $q(h) \propto p(h)e^{-\beta L_S(h)}$, but the interpretation differs.
>
> **Bayesian:** P is a probabilistic model of the target (or of $\mathcal{D}$).
>
> **PAC-Bayes:** P is an inductive bias; the target need not come from P.

Part 1 summary

Five complexity terms, grouped by what they bound:

Flavor	Method	Complexity term
uniform	finite class	$\log\lvert\mathcal{H}\rvert$
uniform	VC theory	$\mathrm{VCdim}(\mathcal{H})$
non-uniform	MDL	$\lvert d(h)\rvert$
non-uniform	SRM over classes	$\mathrm{VCdim}(\mathcal{H}_r) - \log p_r$
non-uniform	PAC-Bayes	$\mathrm{KL}(Q \,\|\, P)$

Uniform methods bound every $h \in \mathcal{H}$ at the same rate; non-uniform methods let the rate depend on the output's complexity.

All five are statistical. None ensures the predictor is efficiently computable.

Efficient learning and proper hardness

Statistical learnability is not the whole story

The Fundamental Theorem says:

$$\text{VCdim}(\mathcal{H}) < \infty \quad \Rightarrow \quad \text{ERM learns statistically.}$$

But ERM is an optimization problem; solving it may be **computationally intractable**.

Efficient learning asks for runtime polynomial in the problem size:

$$\text{poly } (d, 1/\epsilon, \log(1/\delta))$$

where ϵ is the accuracy parameter and δ the failure probability.

Dimension dependence: parametrize by input dimension d and study a family $(\mathcal{H}_d)_{d \geq 1}$.

- $\mathcal{H}_d =$ halfspaces in $\mathbb{R}^d$
- $\mathcal{H}_d =$ conjunctions / 3-term DNFs over d variables

Efficient PAC learning: family setup

For each input length d, let

$$\mathcal{X}_d = \{0,1\}^d, \qquad \mathcal{H}_d \subseteq \{0,1\}^{\mathcal{X}_d}.$$

Definition (efficiently PAC learnable)

A family $(\mathcal{H}_d)_{d\geq 1}$ is **efficiently PAC learnable** if there is a learning rule A such that for every d and every $\epsilon, \delta > 0$, for every realizable $\mathcal{D}$,

$$\mathbb{P}_{S\sim\mathcal{D}^n}[L_{\mathcal{D}}(A(S,d)) \leq \epsilon] \geq 1-\delta,$$

with sample size $n = n(d,\epsilon,\delta) \leq \text{poly}\,(d, 1/\epsilon, \log(1/\delta))$ and runtime polynomial in $d, 1/\epsilon, \log(1/\delta)$.

- **Runtime + sample size.** Both polynomial; statistical learnability only required sample-size bounds.
- **Uniform PAC.** A single A handles every target in every $\mathcal{H}_d$; bounds are the same across targets (contrast with Part 1's non-uniform rates).
- **Realizable setting.** Assumes some $h^* \in \mathcal{H}_d$ has $L_{\mathcal{D}}(h^*) = 0$. Agnostic efficient learning is strictly harder.

What does the algorithm output?

In Part 1, the output was an abstract $\hat{h} \in \mathcal{H}$. Efficient learning also needs to **evaluate** the predictor on new inputs, so the output must be a concrete, computable object.

The learner outputs a string $w \in \{0,1\}^*$ (think: trained weights, a program). A fixed evaluator B computes

$$h_w(x) = B(w, x).$$

Example (halfspaces): $w =$ weight vector; $B(w, x) = \mathbf{1}[w \cdot x \geq 0]$.

Efficiency requires:

- $|w| = \text{poly}\ (d, 1/\epsilon, \log(1/\delta))$ (short description).
- $B(w, x)$ runs in polynomial time (fast prediction).

Examples: what can we learn efficiently?

Halfspaces over $\mathbb{R}^d$.

- Output $w \in \mathbb{R}^d$; $h_w(x) = \mathbf{1}[w \cdot x \geq 0]$.
- Consistent-w condition: $(2y_i - 1)\langle w, x_i \rangle > 0$ for all i; a linear program in w.
- **LP feasibility** is poly in (n, d); with VCdim $= d$ and $n = O(d/\epsilon)$, total runtime is poly $(d, 1/\epsilon, \log(1/\delta))$.

Polynomial thresholds of degree $\leq r$.

- Example: $h(x) = \mathbf{1}[x_1^2 + 2x_1x_2 - x_3 - 1 \geq 0]$.
- Lift $x \mapsto \varphi(x) =$ (all monomials of degree $\leq r$), giving $\binom{d+r}{r}$ features.
- Reduces to **halfspaces over** $\varphi(x) \in \mathbb{R}^{\binom{d+r}{r}}$ poly time for fixed r.

Conjunctions over $\{0,1\}^d$. Our worked example, next.

Each algorithm finds some $h \in \mathcal{H}_d$ consistent with S. We'll abstract this pattern as the FINDCONS **problem.**

Example: conjunctions

Let $\mathcal{H}_d = \mathsf{CONJ}_d$ be conjunctions over $\{0,1\}^d$ with **each variable appearing in at most one literal**:

$$h(x) = \bigwedge_{i \in I_+} x_i \quad \wedge \quad \bigwedge_{i \in I_-} \neg x_i, \quad I_+, I_- \subseteq [d], \quad I_+ \cap I_- = \emptyset.$$

Here $\wedge$ is logical AND: $h(x) = 1$ iff $x_i = 1$ for all $i \in I_+$ and $x_i = 0$ for all $i \in I_-$.

Equivalently, for each variable x_i there are three choices:

- include x_i (i.e. $i \in I_+$),
- include $\neg x_i$ (i.e. $i \in I_-$),
- omit x_i.

Thus

$$|\mathsf{CONJ}_d| = 3^d, \qquad \mathrm{VCdim}(\mathsf{CONJ}_d) = O(d).$$

Elimination algorithm for conjunctions

Input: a labeled sample $S = \left\{\left(x^{(j)}, y^{(j)}\right)\right\}_{j=1}^{n}$ with $x^{(j)} \in \{0,1\}^d$ and $y^{(j)} \in \{0,1\}$.

Algorithm:

1. Start with all $2d$ literals:

$$h = x_1 \wedge \neg x_1 \wedge \cdots \wedge x_d \wedge \neg x_d,$$

   which is identically 0 (every input has some literal failing).

2. For every positive example $(x, 1)$, need $h(x) = 1$, i.e. every surviving literal must be true on x. So **delete the literals violated by x**:
   - if $x_i = 1$, then $\neg x_i$ is false on x; delete $\neg x_i$;
   - if $x_i = 0$, then x_i is false on x; delete x_i.

3. After processing all positives, check that $h(x) = 0$ for every negative example $(x, 0) \in S$.

If yes, output h; otherwise declare inconsistent.

Runtime: $O(nd)$ (step 1 initializes $O(d)$ literals; step 2 scans $O(nd)$ entries; step 3 evaluates h on each negative in $O(d)$), so $\mathsf{FINDCONS}_{\mathsf{CONJ}}$ **is in poly time**.

Why the elimination algorithm works

Claim.

- If some $g \in \mathsf{CONJ}_d$ is consistent with S, the algorithm outputs a consistent h.
- Otherwise it declares inconsistent.

Key observation. After step 2, h is the **maximal conjunction consistent with all positives**:

- every literal true on every positive is kept;
- every other literal is deleted.

Suppose $g \in \mathsf{CONJ}_d$ is consistent with S.

- Each literal of g is true on every positive (else g rejects it).
- By maximality of h, each such literal survives in h, so $\text{literals}(g) \subseteq \text{literals}(h)$.

More literals $\Rightarrow$ more restrictive:

$$h(x) = 1 \quad \Rightarrow \quad g(x) = 1, \quad \text{equivalently,} \quad g(x) = 0 \Rightarrow h(x) = 0.$$

Conclusion.

- g rejects every negative in S, so h does too.
- Step 3 passes; h is consistent.

Why the elimination algorithm works

- If no consistent g exists, step 3 fails and we declare inconsistent.

The consistency problem

Elimination gives efficient learning for CONJ_d by finding a consistent h. What about other classes?

To ask the question across any family $(\mathcal{H}_d)$, name the combinatorial core of what elimination did:

Definition ($\mathsf{FINDCONS}_{\mathcal{H}}$)

Given a labeled sample $S \subseteq \mathcal{X}_d \times \{0,1\}$, either

- return $h \in \mathcal{H}_d$ such that $L_S(h) = 0$, or
- declare that no such h exists.

Pure search problem: no distribution, no ϵ, no δ.

Next slide: efficient FINDCONS gives efficient PAC learning.

Efficient learning from efficient consistency

Efficient consistency implies efficient proper PAC learning

Suppose:
- $\mathrm{VCdim}(\mathcal{H}_d) \leq \text{poly } (d)$,
- $\mathsf{FINDCONS}_{\mathcal{H}}$ is solvable in polynomial time.

Then $(\mathcal{H}_d)$ is efficiently properly PAC learnable in the realizable setting.

Proof sketch.
- Take $n = O((\mathrm{VCdim}(\mathcal{H}_d) + \log(1/\delta))/\epsilon)$ samples; realizability $\Rightarrow \exists h^* \in \mathcal{H}_d, L_S(h^*) = 0$.
- FINDCONS returns $\hat{h} \in \mathcal{H}_d$ with $L_S\left(\hat{h}\right) = 0$; realizable VC bound $\Rightarrow L_{\mathcal{D}}\left(\hat{h}\right) \leq \epsilon$.

Applied to CONJ_d:
- $\mathrm{VCdim}(\mathsf{CONJ}_d) = O(d)$ ✓, elimination solves $\mathsf{FINDCONS}_{\mathsf{CONJ}}$ in $O(nd)$ ✓.
- So CONJ_d is efficiently properly PAC learnable.

Next: $\mathsf{3\text{-}TERM\text{-}DNF}_d$ has $\mathrm{VCdim} = O(d)$ but FINDCONS is NP-hard, so the theorem's hypothesis fails.

3-term DNF

One step beyond conjunctions: take the OR of (up to 3) conjunctions.

Definitions.

- **Term**: a conjunction of literals (so any $T \in \mathsf{CONJ}_d$).
- **DNF** (disjunctive normal form): an OR of terms.
- **3-term DNF:** at most 3 terms. (The constant 3 does not grow with d.)

$$\mathsf{3\text{-}TERM\text{-}DNF}_d = \{T_1 \vee T_2 \vee T_3 : T_i \in \mathsf{CONJ}_d\}.$$

Example: $(x_1 \wedge \neg x_7) \vee (x_2 \wedge x_3 \wedge x_5) \vee (\neg x_4)$.

Sample complexity.

- $|\mathsf{3\text{-}TERM\text{-}DNF}_d| \leq \left(3^d\right)^3 = 3^{3d}$.
- $\mathrm{VCdim}(\mathsf{3\text{-}TERM\text{-}DNF}_d) \leq \log_2 |\mathsf{3\text{-}TERM\text{-}DNF}_d| = O(d)$.

Statistical side is fine, but $\mathsf{FINDCONS}_{\mathsf{3\text{-}TERM\text{-}DNF}}$ is NP-hard.

Proper learning

Every efficient learner we've built (halfspaces, conjunctions) outputs $h_w \in \mathcal{H}_d$. Let's name this property and ask whether it is automatic or a real restriction.

Definition (efficient proper PAC learning)

$(\mathcal{H}_d)_{d\geq 1}$ is **efficiently properly PAC learnable** if there is a learning rule A such that for every d and every $\epsilon, \delta > 0$, for every realizable $\mathcal{D}$,

$$\mathbb{P}_{S\sim\mathcal{D}^n}[L_{\mathcal{D}}(A(S,d)) \leq \epsilon] \geq 1-\delta,$$

with

- sample size $n = n(d,\epsilon,\delta) \leq \mathrm{poly}(d, 1/\epsilon, \log(1/\delta))$,
- runtime polynomial in $d, 1/\epsilon, \log(1/\delta)$,
- **output predictor $h_w \in \mathcal{H}_d$.**

Question. Is poly VC enough for efficient proper PAC learning? Next: no — also need $\mathsf{CONS}_{\mathcal{H}} \in \mathsf{RP}$.

Complexity theory, briefly

- **P:** problems solvable by a poly-time deterministic algorithm. Examples: sorting, shortest path.
- **NP:** "yes"-instances have a short **witness** verifiable in poly time. Examples: SAT (witness = satisfying assignment); $\mathsf{CONS}_{\mathcal{H}}$ (witness = consistent $h \in \mathcal{H}_d$).
- **RP:** randomized version of P. Poly-time algorithm with no false positives and success probability $\geq 1/2$ on yes-instances.
- **NP-hard:** at least as hard as every problem in NP. Conjecturally no poly-time algorithm exists (deterministic or randomized).

Relations: $\mathsf{P} \subseteq \mathsf{RP} \subseteq \mathsf{NP}$. All inclusions conjectured strict; in particular $\mathsf{NP} \neq \mathsf{RP}$ says randomness doesn't crack NP-hard problems.

Proper learning hardness theorem

Decision version of FINDCONS: $\mathsf{CONS}_{\mathcal{H}}(S) = 1 \Leftrightarrow \exists h \in \mathcal{H}_d, L_S(h) = 0.$

$\mathsf{CONS}_{\mathcal{H}} \in \mathsf{RP}$ (randomized polynomial time) means there is a randomized poly-time algorithm that, on input S,

- outputs 0 if $\mathsf{CONS}_{\mathcal{H}}(S) = 0$ (no false positives);
- outputs 1 with probability $\geq 1/2$ if $\mathsf{CONS}_{\mathcal{H}}(S) = 1$.

Learning-to-consistency reduction

If $(\mathcal{H}_d)$ over $\mathcal{X}_d = \{0,1\}^d$ is efficiently properly PAC learnable, then $\mathsf{CONS}_{\mathcal{H}} \in \mathsf{RP}$.

Interpretation.

- A proper learner produces a candidate $h \in \mathcal{H}_d$; checking $L_S(h) = 0$ decides $\mathsf{CONS}_{\mathcal{H}}$.
- Contrapositive: $\mathsf{CONS}_{\mathcal{H}}$ hard $\Rightarrow$ $\mathcal{H}$ not efficiently properly learnable.

Proof of the reduction

Algorithm. Given S of size n:

- run the proper learner on uniform $\mathcal{D}_S$ over S with $\epsilon = 1/(2n)$, $\delta = 1/8$;
- by **properness**, the learner returns some $h \in \mathcal{H}_d$ in polynomial time;
- compute $L_S(h) = (1/n) \sum_{j=1}^{n} \mathbb{1}\left[h\left(x^{(j)}\right) \neq y^{(j)}\right]$; output 1 if $L_S(h) = 0$, else 0.

Inconsistent case ($\mathsf{CONS}_{\mathcal{H}}(S) = 0$):

- No $h \in \mathcal{H}_d$ satisfies $L_S(h) = 0$.
- The learner's output is in $\mathcal{H}_d$ (proper), so $L_S(h) > 0$; the check rejects.
- We always output 0. **No false positives.**

Consistent case ($\mathsf{CONS}_{\mathcal{H}}(S) = 1$):

- $\mathcal{D}_S$ is realizable by $\mathcal{H}_d$, so the PAC guarantee applies: with probability $\geq 7/8$, $L_{\mathcal{D}_S}(h) < 1/(2n)$.
- **Discreteness:** $L_{\mathcal{D}_S}(h) \in \{0, 1/n, 2/n, \ldots\}$, so $< 1/(2n)$ forces $L_{\mathcal{D}_S}(h) = 0$, i.e. $L_S(h) = 0$.
- The check accepts; **success probability** $\geq 7/8$.

Consequence: efficient proper PAC

Combining the two directions we proved:

- **Sufficient** (earlier): poly VCdim + poly-time $\mathsf{FINDCONS}_{\mathcal{H}} \Rightarrow$ efficient proper PAC.
- **Necessary** (this section): efficient proper PAC $\Rightarrow \mathsf{CONS}_{\mathcal{H}} \in \mathsf{RP}$.

Efficient proper PAC: sufficient and necessary conditions

For a family $(\mathcal{H}_d)$:

- If $\mathrm{VCdim}(\mathcal{H}_d) \leq \mathrm{poly}(d)$ and $\mathsf{FINDCONS}_{\mathcal{H}}$ is solvable in polynomial time, then $(\mathcal{H}_d)$ is efficiently properly PAC learnable.
- If $(\mathcal{H}_d)$ is efficiently properly PAC learnable, then $\mathsf{CONS}_{\mathcal{H}} \in \mathsf{RP}$.

- VCdim controls sample size (statistical), while consistency captures the computational obstruction.
- Contrast with the Fundamental Theorem: PAC learnability was characterized by VCdim alone. Efficient PAC adds the computational piece.

Back to 3-term DNF

Pitt-Valiant, 1988

$\mathsf{CONS}_{\text{3-TERM-DNF}}$ is NP-hard (reduction from graph 3-coloring).

Apply the necessary direction to 3-TERM-DNF_d:

- $\mathrm{VCdim}(\text{3-TERM-DNF}_d) = O(d)$ ✓
- $\mathsf{CONS}_{\text{3-TERM-DNF}}$ NP-hard ×

Therefore (assuming NP ≠ RP),

$$\text{3-TERM-DNF}_d \quad \text{is not efficiently properly PAC learnable.}$$

Statistically easy. The hardness theorem assumed $h_w \in \text{3-TERM-DNF}_d$; dropping this requirement (improper learning) sidesteps the theorem.

Improper learning: the remedy

Where did properness bite?

- The reduction's final check $L_S(h) = 0$ decides CONS only because $h \in \mathcal{H}_d$.
- An improper learner could fit S with $h \notin \mathcal{H}_d$ even when no $h \in \mathcal{H}_d$ does.
- The reduction breaks.

So NP-hardness of CONS rules out **proper** efficient learning, but not improper.

Improper learning:

- Drop the constraint $h_w \in \mathcal{H}_d$.
- This was never required by the original efficient PAC definition.

Plan.

- Output h_w from a larger class $\mathcal{H}'_d \supseteq \mathcal{H}_d$ that is efficiently properly learnable.
- For 3-TERM-DNF$_d$: take $\mathcal{H}'_d =$ 3-CNF (next).

Why 3-term DNF is contained in 3-CNF

CNF (conjunctive normal form): an AND of clauses, each clause an OR of literals.

- "3-CNF": every clause has at most 3 literals.
- Mirror of DNF: swap the roles of AND and OR.

Write a 3-term DNF as $T_1 \vee T_2 \vee T_3$ with

$$T_1 = \bigwedge_{i=1}^{r} A_i, \quad T_2 = \bigwedge_{j=1}^{s} B_j, \quad T_3 = \bigwedge_{k=1}^{t} C_k.$$

Distributive law (OR distributes over AND):

$$T_1 \vee T_2 \vee T_3 = \bigwedge_{i,j,k} (A_i \vee B_j \vee C_k).$$

Each clause has exactly 3 literals (one from each term), so every 3-term DNF is a 3-CNF:

$$\mathsf{3\text{-}TERM\text{-}DNF}_d \subseteq \mathsf{3\text{-}CNF}_d.$$

FINDCONS for 3-CNF and why it works

Same template as conjunctions, but over **3-literal clauses** instead of individual literals.

Algorithm.

- Start: F = AND of all $\binom{2d}{3} = O(d^3)$ possible 3-literal clauses.
- For each positive $(x, 1)$, delete every clause falsified by x.
- Check: does $F(x) = 0$ on every negative example?

Runtime. $O(nd^3)$: for each of $O(n)$ examples, scan $O(d^3)$ clauses in $O(1)$ each. Poly in (n, d).

Correctness. After step 2, F is the **maximal 3-CNF consistent with all positives**.

- For any consistent 3-CNF $F^\star$, each clause of $F^\star$ is true on every positive (else $F^\star$ rejects it), so it survives: $\text{clauses}(F^\star) \subseteq \text{clauses}(F)$.
- More clauses $\Rightarrow$ more restrictive: $F^\star(x) = 0 \Rightarrow F(x) = 0$.
- If any consistent $F^\star$ exists, F rejects every negative and the check accepts.

3-TERM-DNF$_d$ is efficiently improperly PAC learnable

Algorithm.

1. Draw n samples from $\mathcal{D}$.
2. In the realizable case, run FINDCONS$_{\text{3-CNF}}$; get a consistent $\hat{h} \in$ 3-CNF$_d$.
3. Output $\hat{h}$ (generally $\notin$ 3-TERM-DNF$_d$, hence improper).

Improper learning of 3-TERM-DNF$_d$ via FINDCONS$_{\text{3-CNF}}$

Assume realizability with target $h^* \in$ 3-TERM-DNF$_d$.

Running FINDCONS$_{\text{3-CNF}}$ takes time poly(n, d) and returns a consistent $\hat{h} \in$ 3-CNF$_d$.

Therefore, with probability $\geq 1 - \delta$,

$$L_{\mathcal{D}}\left(\hat{h}\right) \leq O\left(\frac{\text{VCdim}(\text{3-CNF}_d) + \log(1/\delta)}{n}\right).$$

Sample complexity vs runtime

Rule	Samples	Runtime
proper 3-term DNF	$\tilde{O}(d/\epsilon)$	NP-hard
improper via 3-CNF	$\tilde{O}(d^3/\epsilon)$	poly $(d, 1/\epsilon)$

The improper learner pays more samples to obtain a tractable algorithm.

This is the first major separation between statistical and computational complexity.

Improper hardness

From proper hardness to improper hardness

Part 2 left us with:

- finite VC $\Rightarrow$ statistical learnability;
- NP-hardness of $\mathsf{CONS}_{\mathcal{H}}$ $\Rightarrow$ no efficient **proper** PAC learner;
- 3-term DNF: proper-hard, but improper-tractable via 3-CNF.

Proper hardness only rules out learners whose output lies in $\mathcal{H}$. Allowing output in a larger class (TDNF $\subseteq$ CNF) can make learning easy.

Part 3: when is learning a class hard for every efficient algorithm, no matter what class the output lives in?

Definition.

$$\mathsf{POLY}_d = \{h : \{0,1\}^d \to \{0,1\} : h \text{ computable in time poly } (d)\}.$$

Any efficiently-evaluable class satisfies $\mathcal{H}_d \subseteq \mathsf{POLY}_d$.

Why consider POLY?

- Any efficient learner outputs a predictor evaluable in polynomial time.
- POLY is the class of such predictors: it contains every possible output of every efficient learner.
- Natural benchmark: is every efficiently computable function efficiently PAC learnable?

Valiant's question: is POLY efficiently PAC learnable?

Reduction to consistency. By the learning-to-consistency reduction from Part 2:

$$\mathsf{POLY} \text{ efficiently properly PAC learnable} \quad \Rightarrow \quad \mathsf{CONS}_{\mathsf{POLY}} \in \mathsf{RP} \ .$$

$\mathsf{CONS}_{\mathsf{POLY}} \in \mathsf{NP}$:

- given $h \in \mathsf{POLY}_d$, checking $h(x) = y$ for all $(x, y) \in S$ runs in poly time.

$\mathsf{CONS}_{\mathsf{POLY}}$ is NP-hard:

- reduction from SAT (Pitt-Valiant).

Conclusion: under $\mathsf{NP} \neq \mathsf{RP}$, POLY is not efficiently **properly** PAC learnable.

Valiant's question (improper PAC of POLY) is still open.

Why crypto

Question. Is POLY efficiently **improperly** PAC learnable? (Valiant's question.)

Why NP-hardness doesn't settle it.

- NP-hardness of $\mathsf{CONS}_{\mathsf{POLY}}$ rules out proper learning only.
- Pitt-Valiant reduction needs output $\in$ target class; an improper learner's output need not be.

Strategy: attack a subclass.

- Find $\mathcal{H}_d \subseteq \mathsf{POLY}$ that is not improperly PAC learnable.
- Monotonicity (standard PAC): a PAC learner for POLY handles every target in POLY, hence every target in $\mathcal{H}_d$. So $\mathcal{H}_d$ hard $\Rightarrow$ POLY hard.

Idea from cryptography.

- Construct $\mathcal{H}_d \subseteq \mathsf{POLY}$ from decryption-related functions.
- If $\mathcal{H}_d$ were improperly PAC learnable, one could turn the learner into an attack on the cryptosystem.
- Under the assumed security of the cryptosystem, $\mathcal{H}_d$ is not improperly PAC learnable.

Crypto supplies a hard subclass of POLY, answering Valiant's question negatively.

Ingredients.

- $\mathcal{S}$: plaintext space (messages s);
- $\mathcal{C}$: ciphertext space (ciphertexts c);
- every user has two keys:
  - **public key** K: known to everyone; anyone can encrypt using K;
  - **secret key** K_*: held only by the user; used to decrypt.

A public-key cryptosystem is a pair of algorithms $\left(\mathsf{Enc}_K, \mathsf{Dec}_{K_*}\right)$ with $\mathsf{Enc}_K : \mathcal{S} \to \mathcal{C}$ a bijection (so decryption is well-defined):

- **encryption:** anyone with K computes $c = \mathsf{Enc}_K(s)$ in poly time;
- **decryption:** the user (with K_*) computes $s = \mathsf{Dec}_{K_*}(c)$ in poly time;
- **hardness:** computing s from (K, c) alone (without K_*) is hard.

Example: discrete cube root cryptosystem

Setup.

- $K = pq$ with primes p, q such that $(p-1)(q-1) \bmod 3 \neq 0$ (ensures Enc_K is a bijection);
- secret key $K_* = (p, q)$.

Encryption (easy with K):

$$c = \mathsf{Enc}_K(s) = s^3 \bmod K.$$

Decryption with $K_* = (p, q)$ (poly-time):

- find D with $3D \equiv 1 \pmod{(p-1)(q-1)}$ (extended Euclidean algorithm, poly-time);
- output $\mathsf{Dec}_{K_*}(c) = c^D \bmod K$ (Fermat + CRT give $c^D = s^{3D} \equiv s \pmod K$).

Decryption from K alone:

- must find s satisfying $s^3 \equiv c \pmod K$;
- no known efficient algorithm; conjectured as hard as factoring $K = pq$.

Easy with the secret key, conjectured hard without.

Hardness assumption

Formalizing "hard to decrypt from (K, c) alone":

Hardness assumption (average-case)

No randomized poly-time BREAK satisfies, for every public key K,

$$\mathbb{P}_{s\sim \text{ Unif }(\mathcal{S})}[\text{BREAK }(K, \text{Enc}_K(s)) = s] \geq \frac{1}{\text{poly }(d)}.$$

Why average-case (not worst-case)?

- Real attackers face typical messages, not adversarially chosen ones;
- worst-case hardness could leave typical ciphertexts easy to decrypt, giving no real security;
- standard crypto security is defined on **random** inputs.

Non-negligible. Success probability $\geq 1/\text{poly}(d)$ counts as a break.

Standard security notion; matches PAC's distributional setup.

A hypothesis class from decryption

For each valid key pair (K, K_*), target hypothesis:

$$h_{K,K_*}(c, i) = \mathsf{Dec}_{K_*}(c)[i], \quad \mathcal{H}_d = \left\{ h_{K,K_*} : (K, K_*) \text{ valid of size } d \right\}.$$

- d = security parameter;
- domain: $\mathcal{X}_d = \mathcal{C}_d \times \{1, \ldots, d\}$; labels in $\{0, 1\}$.

Ciphertext distribution $\mathcal{D}_K$ over $\mathcal{C}_d$: draw $s \sim \mathsf{Unif}(\mathcal{S}_d)$, set $c = \mathsf{Enc}_K(s)$.

Training sample. (x, y) with $x = (c, i)$, $c \sim \mathcal{D}_K$, $i \sim \mathsf{Unif}\ \{1, \ldots, d\}$ independent, $y = s[i]$ (realizable: $y = h_{K,K_*}(x)$).

Learner's input. Public key K (not K_*) and an i.i.d. training sample of size n.

Cardinality. $|\mathcal{H}_d| \leq 2^d$, so $\mathrm{VCdim}(\mathcal{H}_d) \leq d$.

Statistically easy (VC $= O(d)$). Hardness is computational.

Learning ⇒ breaking

Assumption. $\mathcal{H}_d$ is efficiently improperly PAC learnable: a poly-time learner A exists.

Breaker. Given K and a fresh challenge $c^* = \mathsf{Enc}_K(s^*)$ (target s^* uniform, unknown), build:

1. **Simulate training data.** Draw s_j uniform, $c_j = \mathsf{Enc}_K(s_j)$, i_j uniform; sample $\big((c_j, i_j), s_j[i_j]\big)$.
2. **Run learner A** with $\epsilon = 1/(4d)$, $\delta = \frac{1}{4}$; obtain $\hat{h}$.
3. **Decode.** Output $\hat{s} = \big(\hat{h}(c^*, 1), \ldots, \hat{h}(c^*, d)\big)$.

PAC guarantee (w.p. $\geq \frac{3}{4}$ over training): $\mathbb{P}_{c \sim \mathcal{D}_K,\, i \sim \mathsf{Unif}}\big[\hat{h}(c, i) \neq s[i]\big] \leq 1/(4d)$.

Union bound over the d bit positions: $\mathbb{P}_{c \sim \mathcal{D}_K}[\text{any bit of } \ s \ \text{ wrong}] \leq d \cdot 1/(4d) = \frac{1}{4}$.

Combined: breaker decodes c^* with prob $\geq \frac{3}{4} \cdot \frac{3}{4} = \frac{9}{16}$.

- Breaker is poly-time (steps 1, 3 poly; A poly by assumption).
- Success $\geq \frac{9}{16}$ is constant, hence $\geq 1/\mathrm{poly}(d)$.
- Violates the crypto hardness assumption ⇒ no efficient PAC learner A exists.

$\mathcal{H}_d$ is not efficiently PAC learnable, even improperly.

Consequences

Under the crypto hardness assumption:

- $\mathcal{H}_d$ is **not** efficiently PAC learnable, even improperly;
- yet $\mathrm{VCdim}(\mathcal{H}_d) \leq d$.

A genuine statistical/computational separation. $O(d)$ samples suffice for an information-theoretic learner; no efficient algorithm matches this.

Part 2 vs Part 3.

- Part 2 (proper hardness): $\mathsf{NP} \neq \mathsf{RP}$ suffices (3-term DNF).
- Part 3 (improper hardness): needs crypto, since improper is harder to rule out.

Lifting to POLY. By monotonicity ($\mathcal{H}_d \subseteq \mathsf{POLY}$, standard PAC), non-learnability of $\mathcal{H}_d$ passes upward: POLY is not efficiently PAC learnable.

Valiant's question: answered negatively under standard crypto. Learning $\neq$ computation, even with unlimited samples.

Landscape of efficient PAC learning

Efficiently properly PAC learnable:

- conjunctions, halfspaces, axis-aligned rectangles.

Efficiently improperly PAC learnable, but not properly:

- 3-term DNF via 3-CNF;
- k-term DNF for constant k via k-CNF.

Not efficiently PAC learnable, even improperly:

- poly-size DNF, intersections of $\omega(\log d)$ halfspaces;
- polynomial-size neural networks;
- poly-time computable functions (POLY).

Efficient PAC learning is a spectrum; representation and assumptions both matter.

Week 4 summary

Week 4 pulled apart three aspects of learning:

- **Approximation.** No-free-lunch forces $\mathcal{H} \subsetneq \{0,1\}^X$. Non-uniform learning replaces one fixed class with a graded bias over a family.
- **Estimation.** MDL, SRM, and PAC-Bayes give non-uniform sample-complexity bounds: the rate depends on the complexity of the output, not just $\mathcal{H}$.
- **Computation.**
  - $\mathsf{CONS}_{\mathcal{H}}$ NP-hard rules out efficient **proper** PAC (e.g., 3-term DNF);
  - crypto hardness rules out even **improper** PAC (Valiant's question, answered negatively for POLY);
  - landscape: properly learnable, improperly-only learnable, not efficiently learnable.

VC dimension characterizes uniform sample complexity. It does not address approximation, non-uniform sample complexity, or computation.