Week 5: Agnostic Learning and Neural Networks

Computational hardness, surrogate losses, and the classical view of neural nets

Tianhao Wang
tianhaowang@ucsd.edu

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

Contents

Bridge from Week 4

Where we left off

Week 4 separated the statistical question from the computational question.

Statistical side:

- $\mathrm{VCdim}(\mathcal{H})$ controls uniform sample complexity;
- MDL, SRM, and PAC-Bayes give non-uniform sample-complexity bounds.

Computational side:

- realizable proper learning asks for a hypothesis that fits the sample;
- some statistically learnable classes are computationally hard;
- allowing improper outputs can help, but crypto hardness shows it does not always help.

Week 4 punchline: finite sample complexity does not imply efficient learning.

The new problem

Last week's empirical primitive was **consistency**:

$$\mathsf{FINDCONS}_{\mathcal{H}}(S): \quad \text{find} \quad h \in \mathcal{H} \quad \text{with} \quad L_S(h) = 0.$$

Justified by the realizable assumption: $\exists h^* \in \mathcal{H}$ with $L_{\mathcal{D}}(h^*) = 0$.

But in practice:

- labels may be noisy;
- the model class may be misspecified.

Then no $h \in \mathcal{H}$ need satisfy $L_S(h) = 0$, and the realizable assumption fails.

In the agnostic setting, the empirical primitive is **empirical risk minimization**:

$$\mathsf{ERM}_{\mathcal{H}}(S): \quad \arg\min_{h \in \mathcal{H}} L_S(h).$$

Population goal: compete with $\inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$.

Realizable to agnostic: the empirical primitive shifts from $\mathsf{FINDCONS}_{\mathcal{H}}$ to $\mathsf{ERM}_{\mathcal{H}}$.

From witness search to optimization

The shift in primitive is also a shift in computational structure.

FINDCONS$_{\mathcal{H}}$

Find $h \in \mathcal{H}$ with $L_S(h) = 0$.

A consistency problem.

ERM$_{\mathcal{H}}$

Return $\arg\min_{h \in \mathcal{H}} L_S(h)$.

An optimization problem.

Even when FINDCONS$_{\mathcal{H}}$ is tractable, ERM$_{\mathcal{H}}$ may be hard.

Examples: halfspaces, conjunctions.

Plan for today

Agnostic ERM

0/1 empirical risk minimization is often computationally hard.

→

Surrogates

Convex losses give tractable optimization for fixed-feature linear models.

→

Neural nets

Learned features increase expressiveness but return us to nonconvex optimization.

The theme is not just "what can represent the truth?" but "what can we optimize, and what bias does optimization impose?"

Agnostic Learning and Surrogate Losses

Realizable PAC learning:

$$\exists h^* \in \mathcal{H} \quad L_{\mathcal{D}}(h^*) = 0,$$

and we ask $L_{\mathcal{D}}(A(S)) \leq \epsilon$.

Agnostic PAC learning: no such assumption. We ask:

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

Estimation error $L_{\mathcal{D}}(A(S)) - \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \leq \epsilon$: an **optimization** goal over $\mathcal{H}$.

Proper agnostic PAC learning

For a family $\mathcal{H}_d \subseteq \{-1,+1\}^{\mathcal{X}_d}$:

Definition (efficient proper agnostic PAC learning)

$\mathcal{H} = (\mathcal{H}_d)$ is **efficiently properly agnostically PAC learnable** if there exist a learner A and a sample-size function

$$n(d,\epsilon,\delta) \le \mathrm{poly}(d, 1/\epsilon, \log(1/\delta))$$

such that for any $d, \epsilon, \delta > 0$ and any $\mathcal{D}$ on $\mathcal{X}_d \times \{-1,+1\}$, with probability $\ge 1-\delta$ over $S \sim \mathcal{D}^n$,

$$L_{\mathcal{D}}(A(S)) \le \inf_{h \in \mathcal{H}_d} L_{\mathcal{D}}(h) + \epsilon \quad \text{and} \quad A(S) \in \mathcal{H}_d,$$

and A runs in time $\mathrm{poly}(d, 1/\epsilon, \log(1/\delta))$.

Proper means the output is still inside the original class.

Same shape as realizable proper PAC, with $\inf_{h \in \mathcal{H}_d} L_{\mathcal{D}}(h)$ in place of 0.

Finite VC handles the statistical part

If $\mathrm{VCdim}(\mathcal{H}_d) \leq \mathrm{poly}(d)$, then ERM achieves

$$L_{\mathcal{D}}\left(\hat{h}\right) \leq \inf_{h \in \mathcal{H}_d} L_{\mathcal{D}}(h) + \epsilon$$

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

$$n = O\left(\frac{\mathrm{VCdim}(\mathcal{H}_d) + \log(1/\delta)}{\epsilon^2}\right)$$

samples.

The remaining question is computational: is $\mathsf{ERM}_{\mathcal{H}}$ efficiently solvable?

Efficient ERM gives efficient agnostic learning

Sufficient conditions for efficient agnostic learning

Suppose for a family $\mathcal{H}_d \subseteq \{-1,+1\}^{\mathcal{X}_d}$:

- $\mathrm{VCdim}(\mathcal{H}_d) \le \mathrm{poly}(d)$;
- every $h \in \mathcal{H}_d$ is efficiently evaluable;
- there is a polynomial-time algorithm for

$$\mathsf{ERM}_{\mathcal{H}}(S) = \arg\min_{h \in \mathcal{H}_d} L_S(h).$$

Then $\mathcal{H}_d$ is **efficiently properly agnostically PAC learnable**.

Proof: uniform convergence + ERM optimality (week 3); ERM oracle gives the polynomial runtime.

Statistical control comes from uniform convergence; computational control comes from the ERM oracle.

From consistency to agreement

For agnostic hardness, we need a decision analogue of $\mathsf{CONS}_{\mathcal{H}}$.

Setting	What the learner must find	Decision problem
Realizable proper	$h \in \mathcal{H}$ with $L_S(h) = 0$	$\mathsf{CONS}_{\mathcal{H}}$
Agnostic proper	$h \in \mathcal{H}$ with small $L_S(h)$	$\mathsf{AGREEMENT}_{\mathcal{H}}$

Agreement problem

$\mathsf{AGREEMENT}_{\mathcal{H}}(S, k) = 1$ iff there exists $h \in \mathcal{H}$ that correctly labels at least k examples in S.

Equivalently: empirical error at most $1 - k/|S|$.

Realizable proper asks for consistency. Agnostic proper asks for empirical error optimization.

Proper agnostic hardness

Learning-to-agreement reduction

If $\mathcal{H}_d$ is efficiently properly agnostically PAC learnable, then

$$\mathsf{AGREEMENT}_{\mathcal{H}_d} \in \mathsf{RP}.$$

Proof. Run the learner on $\mathcal{D}$ = uniform over S with $\epsilon < 1/n$. The agnostic guarantee plus discreteness of L_S force $\hat{h}$ to be an exact empirical minimizer.

Corollary. If $\mathsf{AGREEMENT}_{\mathcal{H}_d}$ is NP-hard and $\mathsf{NP} \neq \mathsf{RP}$, then **$\mathcal{H}_d$ is not efficiently properly agnostically PAC learnable**.

Same shape as week 4: efficient proper learning $\Rightarrow$ decision problem in RP.

Two directions for proper agnostic learning

Combining the two directions:

- polynomial VC dimension plus polynomial-time $\mathsf{ERM}_{\mathcal{H}_d}$ gives efficient proper agnostic learning;
- efficient proper agnostic learning implies $\mathsf{AGREEMENT}_{\mathcal{H}_d} \in \mathsf{RP}$.

Already hard in the realizable setting (Week 4):

- poly-time functions;
- poly-length logical formulas;
- polynomial-size neural networks.

New agnostic proper hardness:

- halfspaces over $\{0,1\}^d$: $\mathsf{FINDCONS}_{\mathcal{H}_d}$ easy, $\mathsf{AGREEMENT}_{\mathcal{H}_d}$ NP-hard;
- conjunctions over $\{0,1\}^d$: $\mathsf{FINDCONS}_{\mathcal{H}_d}$ easy, $\mathsf{AGREEMENT}_{\mathcal{H}_d}$ NP-hard.

Assuming $\mathsf{NP} \neq \mathsf{RP}$, neither class is efficiently properly agnostically PAC learnable.

Positive example: unions of intervals

Class. $\mathcal{H}_k$ = unions of $\leq k$ intervals on $\mathbb{R}$: $h(x) = +1$ iff $x \in \bigcup_{j=1}^{k} [a_j, b_j]$.

$\mathbf{VCdim(\mathcal{H}_k) = 2k}$. A union of k intervals creates at most k "+1-blocks" along any sorted sequence:

- shatters $2k$ points (any labeling has $\leq k$ +1-blocks);
- fails on $2k + 1$ (alternating $+, -, +, \ldots, +$ needs $k + 1$ blocks).

Why $\mathbf{ERM_{\mathcal{H}_k}}$ is tractable. Sort the sample so $x_1 < x_2 < \ldots < x_n$.

- A union of k intervals labels the sorted sample as a $+1/-1$ sequence with $\leq k$ blocks of $+1$'s.
- ERM reduces to: find the best $\leq k$-block labeling matching $(y_1, \ldots, y_n)$. A 1D combinatorial problem, solvable in $O(nk)$ time.

Therefore $\mathcal{H}_k$ is efficiently properly agnostically PAC learnable.

Sortable 1D structure reduces $\mathsf{ERM}_{\mathcal{H}_k}$ to a poly-time combinatorial problem $\Rightarrow$ efficient agnostic PAC.

Halfspaces: realizable vs agnostic

Realizable case: find w with $y_i \langle w, x_i \rangle \geq 1$ for all i.

- n linear constraints in $w \Rightarrow$ a **single linear program**. Poly-time.

Agnostic case: $\min_w \sum_i \mathbf{1}[y_i \langle w, x_i \rangle \leq 0]$.

- Each w misclassifies a subset $M(w) \subseteq [n]$; ERM minimizes $|M(w)|$.
- Equivalently: choose a mistake set M, then check whether some w correctly classifies $[n] \setminus M$ (an LP per fixed M).
- Choice of M: 2^n subsets, **no 1D order or decomposability** to exploit (unlike unions of intervals).

Realizable: one LP. Agnostic: a hidden combinatorial search over the mistake set.

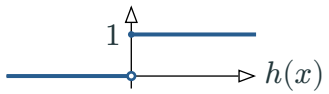
Source of the hardness: nonconvex 0/1 loss

Halfspace agnostic ERM:

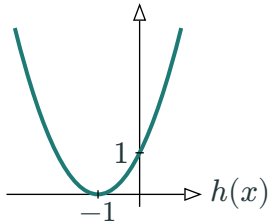
$$\min_{w \in \mathbb{R}^d} \sum_{i=1}^{n} \ell(h_w(x_i), y_i), \quad h_w(x) = \langle w, x \rangle,$$

with $\ell_{0/1}(h(x), y) = \mathbf{1}[yh(x) \leq 0]$.

$\ell_{0/1}(h(x), y = -1)$



$\ell_{\text{sqr}}(h(x), y = -1) = (h(x) + 1)^2$



As a function of w, $\sum_i \ell_{0/1}(\langle w, x_i \rangle, y_i)$ is **piecewise constant**: it jumps each time some $y_i \langle w, x_i \rangle$ crosses 0 but is flat between, with 2^n **flat regions** (one per mistake set).

The 0/1 loss is nonconvex: discontinuous and piecewise constant in w, hard to optimize.

Surrogate loss idea

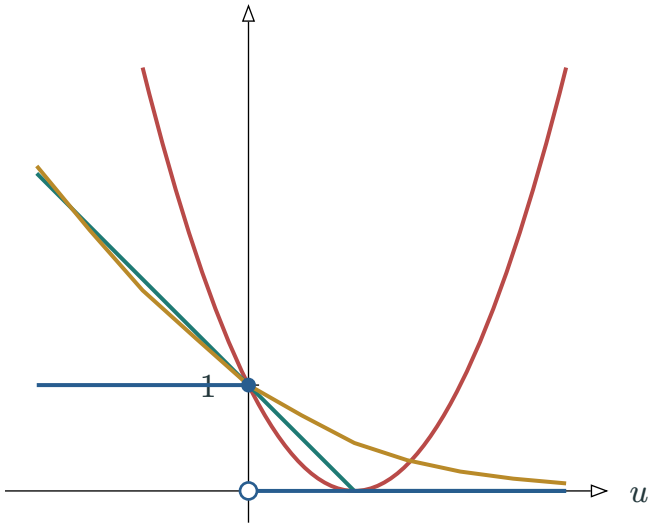
Replace $\ell_{0/1}$ with a surrogate $\ell(z, y)$ that is:

- **convex in z**
- upper-bounds $\ell_{0/1}$: $\ell_{0/1}(z, y) \le \ell(z, y)$.

Common choices (in margin $u = yh(x)$):

- **0/1**: $\mathbf{1}[u \le 0]$
- **hinge**: $(1 - u)_+$
- **logistic**: $\log(1 + e^{-u}) / \log 2$
- **square**: $(1 - u)^2$

All cross at $(0, 1)$; surrogates upper-bound $\ell_{0/1}$.



Surrogates turn the hard 0/1 problem into a tractable convex one; minimizing them aims to control classification error.

Hinge loss ERM is tractable

For linear predictors $h_w(x) = \langle w, x \rangle$:

$$\min_w \frac{1}{n} \sum_i (1 - y_i \langle w, x_i \rangle)_+$$

is **convex** in w.

It can be written as a linear program:

$$\min_{w,\xi} \sum_i \xi_i$$

$$y_i \langle w, x_i \rangle \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

Surrogates turn a hard empirical $0/1$ problem into a tractable convex problem.

Did we change the learning problem?

With $h_w(x) = \langle w, x \rangle$ as the linear predictor, the computational move is

$$\min_w L_S^{0/1}(h_w) \quad \Rightarrow \quad \min_w L_S^{\text{hinge}}(h_w).$$

Since $\ell_{0/1}(z, y) \leq \ell(z, y)$, every score function h satisfies

$$L_S^{0/1}(h) \leq L_S^{\ell}(h),$$

where $L^{0/1}$ is evaluated on $\mathsf{sign}(h(x))$.

But **minimizing an upper bound need not minimize the quantity being bounded**.

We swapped the empirical objective for an upper bound; the question is whether minimizing the bound also minimizes the original.

Realizable case: no loss on the sample

Suppose the sample is linearly separable:

$$\exists w^* \quad \forall i, \quad y_i \langle w^*, x_i \rangle > 0.$$

Rescale. Hinge loss vanishes only at margin ≥ 1; a small-margin w^* can still have positive hinge loss.

- Let $\gamma = \min_i y_i \langle w^*, x_i \rangle > 0$ and $\tilde{w} = w^*/\gamma$.
- Smallest margin under $\tilde{w}$ is exactly 1 (valid since sign is scale-invariant).
- $y_i \langle \tilde{w}, x_i \rangle \geq 1$ for every i, so $L_S^{\text{hinge}}(h_{\tilde{w}}) = 0$.

Hinge ERM.

- $\min_w L_S^{\text{hinge}}(h_w) = 0$.
- Any hinge ERM $\hat{h}$ satisfies $L_S^{\text{hinge}}\left(\hat{h}\right) = 0$.
- Since $\ell_{0/1} \leq \ell_{\text{hinge}}$, also $L_S^{0/1}\left(\hat{h}\right) = 0$.

In the separable case, the surrogate preserves consistency on the training sample.

Non-realizable case: what remains?

Setup.

- $\mathcal{F}$: real-valued score class; 0/1 evaluated on $\mathsf{sign}(h(x))$.
- Surrogate bounded: $\ell_{0/1}(h(x), y) \le \ell(h(x), y) \le B$.
- Uniform convergence: w.p. $\ge 1 - \delta$, $\sup_{h \in \mathcal{F}} \left| L^{\ell}_{\mathcal{D}}(h) - L^{\ell}_{S}(h) \right| \le \epsilon$.

Surrogate excess-risk bound. Let $\hat{h} \in \arg\min_{h \in \mathcal{F}} L^{\ell}_{S}(h)$ and $h^* \in \arg\min_{h \in \mathcal{F}} L^{\ell}_{\mathcal{D}}(h)$. Then:

$$
\begin{aligned}
L^{0/1}_{\mathcal{D}}\left(\hat{h}\right) &\le L^{\ell}_{\mathcal{D}}\left(\hat{h}\right) && \left(\ell_{0/1} \le \ell\right) \\
&\le L^{\ell}_{S}\left(\hat{h}\right) + \epsilon && \text{(uniform convergence)} \\
&\le L^{\ell}_{S}(h^*) + \epsilon && \text{(ERM optimality)} \\
&\le L^{\ell}_{\mathcal{D}}(h^*) + 2\epsilon && \text{(uniform convergence)} \\
&= \inf_{h \in \mathcal{F}} L^{\ell}_{\mathcal{D}}(h) + 2\epsilon && \text{(definition of } h^* \text{)}.
\end{aligned}
$$

In the non-realizable case, the comparator is surrogate risk.

Comparator caveat

The bound compares to the **best surrogate**, not the best 0/1 classifier:

$$\inf_{h \in \mathcal{F}} L^{\ell}_{\mathcal{D}}(h), \quad \text{not} \quad \inf_{h \in \mathcal{F}} L^{0/1}_{\mathcal{D}}(h).$$

Why they differ.

- 0/1 only counts mistakes; surrogates also penalize correct-but-low-margin predictions.
- Hinge-best h may sacrifice an outlier for bulk margin.
- 0/1-best h would never make that trade.
- Equality only in special cases (e.g., realizable data: $\inf L^{0/1} = \inf L^{\text{hinge}} = 0$).

Bridging the gap (calibration).

- When does the surrogate's population minimizer match the sign of the 0/1-optimal predictor?
- Hinge, logistic, square: calibrated under mild conditions.
- Out of scope here.

The surrogate gives a tractable objective; whether its minimizer is also a good 0/1 classifier needs an additional argument.

Convex functions and sets

Convex function. $f : \mathbb{R}^d \to \mathbb{R}$ is convex if for all $u, v \in \mathbb{R}^d$ and $\lambda \in [0, 1]$,

$$f(\lambda u + (1 - \lambda)v) \leq \lambda f(u) + (1 - \lambda) f(v).$$

Geometrically: the chord between $(u, f(u))$ and $(v, f(v))$ lies above the graph.

Convex set. $\mathcal{C} \subseteq \mathbb{R}^d$ is convex if for all $u, v \in \mathcal{C}$ and $\lambda \in [0, 1]$, $\lambda u + (1 - \lambda)v \in \mathcal{C}$.

Why we care. For a convex objective on a convex set, **every local minimum is a global minimum**, and gradient methods reach it.

Affine map. $A : \mathbb{R}^d \to \mathbb{R}^k$ is affine if $A(u) = Mu + b$ for some matrix M and vector b. (Linear plus a constant shift.)

Three operations preserving convexity (the ones we will use).

- sum: f, g convex $\Rightarrow f + g$ convex;
- affine composition: f convex, A affine $\Rightarrow f \circ A$ convex;
- pointwise max: f, g convex $\Rightarrow \max(f, g)$ convex (e.g., $(z)_+ = \max(0, z)$).

What made hinge ERM convex?

We want $L_S^{\text{hinge}}(w) = \frac{1}{n}\sum_i (1 - y_i h_w(x_i))_+$ convex in w. What shape must h_w take?

Sufficient: h_w affine in w. The three rules compose:

- $z \to (1 - y_i z)_+$ is convex in z (rule 3, max of two affines);
- $(1 - y_i h_w(x_i))_+$ is convex in w (rule 2: convex $\circ$ affine);
- $L_S^{\text{hinge}}(w)$ is convex (rule 1: average of convex).

Necessary: affine is the only choice. For one hinge term:

- if $y_i = +1$: $(1 - h_w(x_i))_+$ convex in $w \Rightarrow h_w(x_i)$ **concave** in w;
- if $y_i = -1$: $(1 + h_w(x_i))_+$ convex in $w \Rightarrow h_w(x_i)$ **convex** in w.

Both signs appear in S, so $h_w(x_i)$ is both convex and concave in w, hence affine:

$$h_w(x) = \langle w, \varphi(x) \rangle + c.$$

Convex learning template

The hinge-loss argument generalizes to any convex surrogate. The convex learning template:

$$h_w(x) = \langle w, \varphi(x) \rangle, \quad w \in \mathcal{W},$$

where:

- $\varphi : \mathcal{X} \to \mathbb{R}^p$ is fixed before optimizing w (absorb the constant by appending 1 to φ);
- $\mathcal{W}$ is a convex feasible set;
- for each label y, the map $z \to \ell(z, y)$ is convex.

Then by the **same three rules**, the empirical surrogate risk

$$L_S^\ell(w) = \frac{1}{n} \sum_i \ell(\langle w, \varphi(x_i) \rangle, y_i)$$

is convex in w.

Convex learning means fixed features plus learned linear weights.

Examples of the template

Two axes of variation, both keeping the problem convex:

Vary the features φ (fixed before optimizing w):
- linear: $\varphi(x) = x$;
- polynomial features of fixed degree;
- kernels / SVMs: possibly high-dimensional φ.

Vary the loss ℓ (convex in the score):
- hinge $\to$ SVM;
- logistic $\to$ logistic regression;
- square $\to$ least squares.

- **Learned**: the top linear weights w.
- **Not learned**: the feature map φ itself.

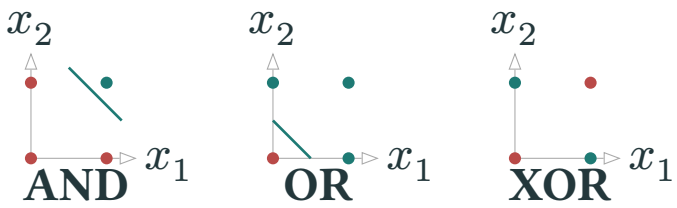
Convexity is bought by deciding the representation before the optimization problem begins.

What can a single linear predictor represent?

Take $\varphi(x) = x$ on $x \in \{0,1\}^d$, so $h(x) = \mathsf{sign}(\langle w, x\rangle + b)$. With $d = 3$:

- **AND**: $x_1 \wedge x_2 \wedge x_3 = \mathsf{sign}(x_1 + x_2 + x_3 - 2.5)$ (works)
- **OR**: $x_1 \vee x_2 \vee x_3 = \mathsf{sign}(x_1 + x_2 + x_3 - 0.5)$ (works)
- **XOR (parity)**: $x_1 \oplus x_2 = ???$ (no separating hyperplane)

In 2D: AND, OR live on one side of a line; XOR's positive points $(0,1), (1,0)$ are diagonal.



Linear predictors capture half-spaces of the raw inputs. Whether the task lives in such a half-space is decided by φ, not by the learner.

Parities: statistically easy, linearly hard

Generalize XOR. For $I \subseteq [d]$ define

$$\chi_{I(x)} = (-1)^{\sum_{i \in I} x_i}, \quad x \in \{0,1\}^d,$$

the **parity class** $\mathsf{PARITIES}_d$ with $|\mathsf{PARITIES}_d| = 2^d$.

Statistically easy. $\mathrm{VCdim}(\mathsf{PARITIES}_d) = d$, so $\sim d$ samples suffice (uniform convergence).

Linearly hard. If a fixed $\varphi : \{0,1\}^d \to \mathbb{R}^D$ realizes every parity as

$$\chi_{I(x)} = \langle w_I, \varphi(x) \rangle \quad \text{for some } w_I,$$

then $D \geq 2^d$.

- **Intuition.** XOR needed a 2-bit interaction feature ($x_1 x_2$). A k-bit parity needs a k-bit interaction feature. Realizing **every** subset I means a feature for every subset of bits, and there are 2^d of them.

Linear/convex learning with a fixed φ has a representational ceiling: a statistically tractable class can demand exponential feature dimension.

Part 1 summary

Three takeaways from Part 1:

- **Agnostic hardness.** Even when realizable learning is tractable, agnostic proper learning can be hard.
- **Convex surrogates as a workaround.** Replace 0/1 with a convex upper bound plus an affine predictor; the guarantee is against **surrogate risk**.
- **Convex learning has a ceiling.** Convexity forces φ to be fixed before optimization; some statistically tractable classes (parities) then require **exponential** feature dimension.

> Next: what changes when φ is learned along with w?

Neural Networks: The Classical View

A 2-unit fix for XOR

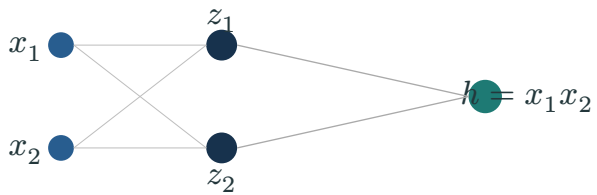
Part 1 ended with the parity ceiling: no fixed φ of polynomial dimension realizes all parities. The simplest case, XOR ($x_1 x_2$ on $\{-1, +1\}^2$), is already not a single halfspace.

Two learned halfspace features fix it:

$$z_1 = \mathsf{sign}(x_1 + x_2 - 1),$$

$$z_2 = \mathsf{sign}(-x_1 - x_2 - 1),$$

$$h(x) = \mathsf{sign}(z_1 + z_2 + 1) = x_1 x_2.$$



A hidden layer creates a learned nonlinear feature: h is linear in z, not in x.

The transition to neural networks

Neural networks move the representation into the learning problem.

Fixed features (Part 1)

Feature engineering. Pick φ first; learn only w.

$$h_w(x) = \langle w, \varphi(x) \rangle.$$

Convex in w.

Learned features (Part 2)

Deep learning. Parameterize φ_u; learn (u, w) jointly.

$$f_{u,w}(x) = \langle w, \varphi_u(x) \rangle.$$

In general nonconvex in (u, w).

For fixed u, the top layer is still linear in w. Jointly in (u, w), the score is not affine.

$$\text{fixed } \varphi \quad \Rightarrow \quad \text{convex optimization over } w$$

$$\text{learned } \varphi_u \quad \Rightarrow \quad \text{representation learning}$$

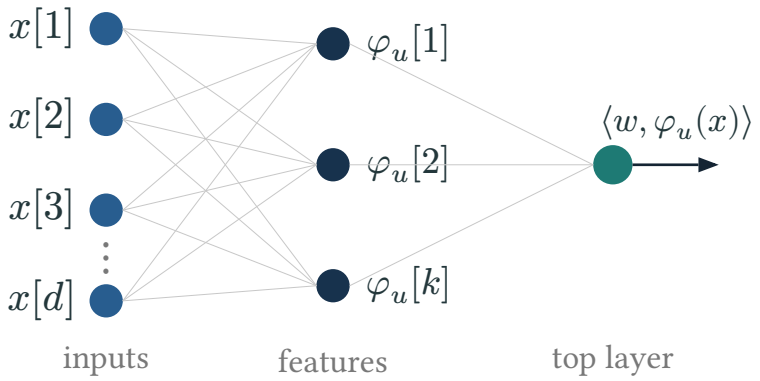
Neural nets as learned features

Concrete form for a one-hidden-layer network with activation σ:

$$\varphi_u(x) = (\sigma(\langle u_1, x\rangle), \ldots, \sigma(\langle u_k, x\rangle)), \quad f_{u,w}(x) = \langle w, \varphi_u(x)\rangle.$$

- $u_1, \ldots, u_k$ define the hidden features.
- w combines those features linearly.
- Learning u means learning the representation.

This is the fixed-feature template with φ replaced by φ_u.



Neural networks learn the feature map and the linear predictor together.

What the tradeoff asks us to study

Three questions to ask about neural networks:

Expressive power

What functions can learned features represent?

Statistics

How many samples are needed to learn the weights?

Computation

Can we find good weights efficiently?

Classical theory's preview:

- **Representation**: very powerful.
- **Statistics**: sample complexity can scale with parameter count.
- **Computation**: hard in the worst case.

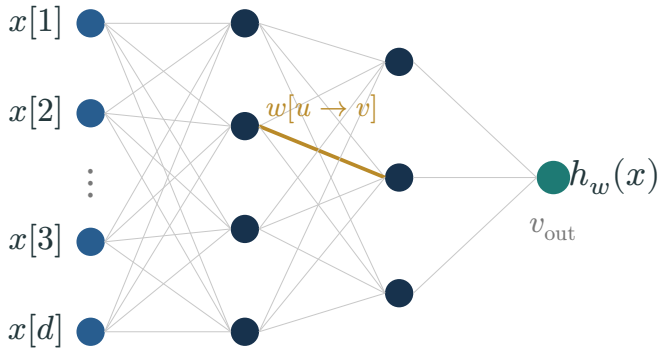
NN gain representational power at the cost of convex optimization. The next sections work out the price.

Feed-forward networks: DAG view

For a node v with predecessors $u \to v$:

$$a[v] = \sum_{u \to v \in E} w[u \to v] o[u],$$

$$o[v] = \sigma(a[v]).$$



- Input nodes: $o[v_i] = x[i]$.
- Output: $h_w(x) = o[v_{\text{out}}]$.
- Bias by adding a constant input $o[v_0] = 1$.

- **Architecture:** directed acyclic graph $G(V, E)$ plus activation σ.
- **Parameters:** one weight $w[u \to v]$ per edge, $|E|$ in total.
- **Evaluation:** compute nodes in a topological order; depth is the longest input-output path.

The DAG view is flexible: it includes fully connected, sparse, and skip-connection architectures.

Feed-forward networks: layer-wise view

Fully connected feed-forward network with L hidden layers and widths $d_0, d_1, \ldots, d_L, d_{L+1}$, where $d_0 = d$ (inputs) and d_{L+1} is the output dimension:

$$z_0 = x, \quad z_\ell = \sigma(W_\ell z_{\ell-1} + b_\ell), \quad h_w(x) = W_{L+1} z_L + b_{L+1}.$$

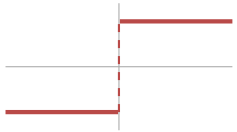
- **Sizes:** $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$, $b_\ell \in \mathbb{R}^{d_\ell}$.
- **Depth:** L hidden layers, $L+1$ weight matrices.
- **Parameter count:** $|w| = \sum_{\ell=1}^{L+1} (d_\ell d_{\ell-1} + d_\ell)$, dominated by the W_ℓ matrices.
- **Output convention:** no activation on the final layer; sign or other output rule is applied separately.

Special case ($L = 1$). Recovers the one-hidden-layer NN: $\varphi_u(x) = \sigma(W_1 x + b_1)$ and $h_w(x) = W_2 \varphi_u(x) + b_2$.

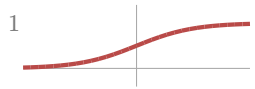
Full parameter: $w = (W_1, b_1, \ldots, W_{L+1}, b_{L+1})$.

Activation functions

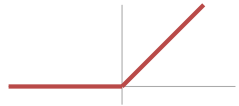
$\mathsf{sign}(z) = +1 \text{ if } z \geq 0, \text{ else } -1$



$\mathrm{sigmoid}(z) = 1/(1+e^{-z})$



$\mathsf{ReLU}(z) = \max(0, z)$



- **Sign as a limit**: $\mathrm{sigmoid}(\beta z) \to \mathsf{sign}(z)$ as $\beta \to \infty$. So a steep sigmoid approximates a threshold.
- **Equivalent for representation**: continuous activations (ReLU, sigmoid) realize the same hypothesis class up to approximation; e.g. two ReLU units exactly simulate one ramp, and a steep sigmoid approximates a sign unit.

Choice of σ matters strongly for optimization and smoothness; many representation statements are robust to the choice.

The neural network hypothesis class

Once we fix an architecture G and an activation σ, varying the weights gives a hypothesis class:

$$\mathcal{H}(G, \sigma) = \left\{ h_w : w \in \mathbb{R}^{|E|} \right\}.$$

- **Fixed before training:** the DAG $G(V, E)$ and the activation σ.
- **Learned from data:** the weight vector w, one entry per edge.

Three questions about $\mathcal{H}(G, \sigma)$:

Expressive power	approximation error	What functions does $\mathcal{H}(G, \sigma)$ represent or approximate?
Sample complexity	estimation error	How many examples to learn w?
Computation	optimization error	Can good w be found efficiently?

A neural network is a parametric family. The next sections answer these three questions.

Expressive power: threshold gates

Use the convention $\mathsf{sign}(t) = +1$ when $t \geq 0$ and -1 otherwise.

For input $x \in \{-1, +1\}^d$, a threshold unit computes $\mathsf{sign}(\langle w, x \rangle + b)$.

Three building blocks, each implementable by a single unit:

- **NOT** of x_j: take $w = -e_j$, $b = 0$.
- **AND** of $x_{j_1}, \ldots, x_{j_k}$: take $w = e_{j_1} + \ldots + e_{j_k}$, $b = -k + 1$.

  Output is $+1$ iff every selected coordinate equals $+1$.
- **OR** of $x_{j_1}, \ldots, x_{j_k}$: take $w = e_{j_1} + \ldots + e_{j_k}$, $b = k - 1$.

  Output is $+1$ iff at least one selected coordinate equals $+1$.

A single threshold unit is already a logical gate. {NOT, AND, OR} is functionally complete: every Boolean function builds from these.

Small circuits as networks

Let $\mathsf{CIRCUIT}_d(s, L)$ be Boolean functions on d inputs computable by AND/OR/NOT circuits with:

- **size** $\leq s$ (number of gates);
- **depth** $\leq L$ (longest input-to-output gate path).

Network architecture. $G_{s,L}$ = layered threshold network of depth L with at most s threshold units (one per circuit gate).

Inclusion. Compose the gate constructions from the previous slide layer by layer:

$$\mathsf{CIRCUIT}_d(s, L) \subseteq \mathcal{H}\big(G_{s,L}, \mathsf{sign}\big).$$

Proof idea.

- put circuit gates of depth ℓ into network layer ℓ;
- use weights $+1$ or -1 for wires and negated wires, 0 for absent wires;
- use the AND/OR biases from the previous slide.

A circuit with s gates and depth L becomes a threshold network with $|V| = O(s)$ units and depth L.

Any Boolean function via DNF

Every $f : \{-1, +1\}^d \to \{-1, +1\}$ has a **DNF** representation: an OR of ANDs, one for each input $a \in \{-1, +1\}^d$ with $f(a) = +1$.

Let $P = \{a \in \{-1, +1\}^d : f(a) = +1\}$ be the set of **positive inputs**.

One AND per $a \in P$. The neuron

$$z_a(x) = \mathsf{sign}\left(\sum_{j=1}^{d} a_j x_j - d + 1\right)$$

has weight $w = a$, bias $b = -d + 1$, and $z_a(x) = +1$ **iff** $x = a$.

DNF as a depth-2 threshold network:

- hidden layer: one z_a per positive input $a \in P$;
- output unit: OR over the hidden units (also a threshold unit).

Every Boolean function on $\{-1, +1\}^d$ is realized by a depth-2 threshold network: hidden = ANDs detecting positive inputs, output = OR.

DNF construction size

Put one hidden unit z_a for each $a \in P$ and OR their outputs:

$$h(x) = \mathsf{sign}\left(\sum_{a \in P} z_a(x) + |P| - 1\right).$$

Three layers: d inputs, $|P|$ hidden units, 1 output. Since $|P| \leq 2^d$:

- hidden units: $|P| \leq 2^d$;
- vertices: $d + |P| + 1 \leq 2^d + d + 1$;
- edges: $d\ |P|$ (input to hidden) $+\ |P|$ (hidden to output) $= (d+1)|P| \leq (d+1) \cdot 2^d$.

This is a lookup table built into the network: one neuron per positive input. It proves expressiveness, not learnability.

Universal representation is possible with exponential size.

What can small networks represent?

DNF gives universal representation but $|V| = 2^d$. Many useful functions need **far smaller** networks.

Intersection of k halfspaces (1 hidden layer, AND output):

- hidden: $z_j = \mathsf{sign}\big(\langle u_j, x\rangle + b_j\big)$ for $j = 1, \ldots, k$;
- output: $\mathsf{AND}(z_1, \ldots, z_k)$.

Union of intersections of halfspaces (2 hidden layers):

- hidden 1: halfspaces;
- hidden 2: AND over each intersection;
- output: OR over intersections.

Feature-learning view.

- each hidden unit: linear combination, then sign;
- depth L stacks L rounds of **compositional feature learning**.

Universal representation: time-T

Let $\mathsf{TIME}_d(T)$ = Boolean functions $f : \{-1,+1\}^d \to \{-1,+1\}$ computable by an algorithm (e.g., Turing machine) in at most T steps.

Circuit simulation (classical). A T-step computation unrolls into an AND/OR/NOT circuit of depth $O(T)$ and width $O(T)$ (so $O(T)$ gates).

Network embedding. By the circuit-to-network construction:

$$\mathsf{TIME}_d(T) \subseteq \mathcal{H}(G_T, \mathsf{sign})$$

with G_T of depth $O(T)$, width $O(T)$, and $|E| = O(T) \cdot O(T)^2 = O(T^3)$ (fully-connected layers).

Poly-time computable $\Rightarrow$ poly-size network.

Capacity: parameter-count bounds

Representation answered. Now the second question: how many samples to learn the parameters?

Let E = number of scalar parameters (edges of G), L = depth.

VC scales with the parameter count E (up to log/depth factors):

- threshold activation ($\sigma = \mathsf{sign}$, output ± 1): $\mathrm{VCdim}(\mathcal{H}(G, \mathsf{sign})) = O(E \log E)$;
- piecewise-linear ($\sigma = \mathsf{ReLU}$): $\mathrm{VCdim}(\mathcal{H}(G, \mathsf{ReLU})) \leq O(EL \log E)$;
- r-bit weights (each weight stored in r bits, so $|\mathcal{H}| \leq 2^{Er}$): $\log|\mathcal{H}(G, \sigma)| \leq O(Er)$.

Activation regularity matters. With oscillatory σ (e.g. sin), **a single unit** can have **infinite VCdim**.

For well-behaved activations, sample complexity scales with the parameter count E.

Capacity implies sample complexity

Consider 0/1 loss. Let $\hat{h} \in \arg\min_{h \in \mathcal{H}} L_S(h)$ (ERM) and $h^* \in \arg\min_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$ (best in class).

Uniform convergence. With probability $1 - \delta$, for all $h \in \mathcal{H}(G, \sigma)$:

$$|L_{\mathcal{D}}(h) - L_S(h)| \leq \epsilon$$

once $n = O((\mathrm{VCdim}(\mathcal{H}(G, \sigma)) + \log(1/\delta))/\epsilon^2)$.

Agnostic ERM chain (uniform convergence at $\hat{h}$, then ERM, then uniform convergence at h^*):

$$L_{\mathcal{D}}\left(\hat{h}\right) \leq L_S\left(\hat{h}\right) + \epsilon \leq L_S(h^*) + \epsilon \leq L_{\mathcal{D}}(h^*) + 2\epsilon.$$

Hence $L_{\mathcal{D}}\left(\hat{h}\right) \leq L_{\mathcal{D}}(h^*) + 2\epsilon$ (agnostic PAC).

Sample complexity scales with the VC dimension.

Universal learning: the sample side

Specialize to the universal time-T class ($E = O(T^3)$, $L = O(T)$, piecewise-linear σ).

VC bound:

$$\mathrm{VCdim}(\mathcal{H}(G_T, \sigma)) \le O(EL \log E) = O(T^3 \cdot T \cdot \log T) = \boldsymbol{O(T^4 \log T)}.$$

Sample complexity (plug into the previous slide):

$$n = \mathbf{poly}(T, 1/\epsilon, \log(1/\delta)).$$

Parallel with universal representation.

- representation: every $f \in \mathsf{TIME}_d(T)$ fits in a poly-size network;
- learning: that network is learnable from poly-many samples.

Statistical universality holds. But this assumes we can find $\hat{h}$ over a huge nonconvex class.

Optimization is nonconvex

Training a network means solving

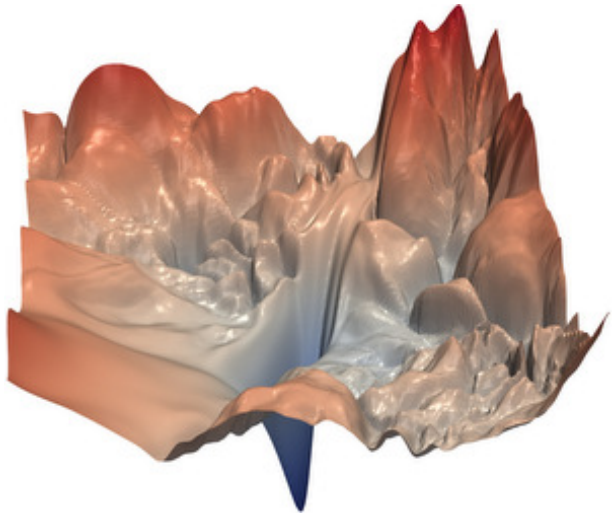
$$\mathsf{ERM}(S) = \arg\min_{w} L_S(h_w).$$

This problem is **generally nonconvex**:

- even when the loss is convex in the score $h_w(x)$;
- even when σ is piecewise linear (ReLU);
- even in the realizable case (some network fits S exactly).

Why?

- bilinearity: even with linear σ, $h_w(x) = \langle w_2, W_1 x \rangle$ is bilinear in (W_1, w_2);
- symmetries: $f_{u,w} = f_{-u,-w}$ gives multiple equivalent minima.



ResNet-56 (no skip), Li et al. 2018

Nonconvexity is intrinsic to learning features, not an artifact of activation or loss.

ERM is NP-hard, already for tiny networks

Recall (Part 1). For a class $\mathcal{H}$, NP-hardness of the decision problems

- $\mathsf{CONS}_{\mathcal{H}}(S)$: $\exists h \in \mathcal{H}$ with $L_S(h) = 0$?
- $\mathsf{AGREEMENT}_{\mathcal{H}}(S, k)$: $\exists h \in \mathcal{H}$ correctly labeling $\geq k$ examples?

rules out efficient **proper realizable** (resp. **proper agnostic**) learning, assuming $\mathsf{NP} \neq \mathsf{RP}$.

Apply to neural networks. Let $\mathcal{H}_d = \mathcal{H}(G, \mathsf{sign})$ with one hidden layer of just **2 units**, inputs $x \in \{-1, +1\}^d$.

Theorem (Blum, Rivest 1992):

- $\mathsf{CONS}_{\mathcal{H}_d}$ is **NP-hard**
- $\mathsf{AGREEMENT}_{\mathcal{H}_d}$ is **NP-hard**.

Same hardness for surrogate losses (hinge: Bartlett-Ben-David 2002; squared: Šíma 2002).

"Data generated by a 2-unit net" is not enough for efficient proper learning. The wall is computational, not statistical.

Even improper learning is hard

Escape hatch? Drop the proper-learning constraint:

- **proper:** output $h \in \mathcal{H}_d$;
- **improper:** output any efficiently evaluable predictor.

Why this still fails:

- recall $\mathsf{TIME}_d(T) \subseteq \mathcal{H}(G_T, \mathsf{sign})$ (universal-representation slide);
- so efficient improper learning of $\mathcal{H}(G_T, \mathsf{sign})$ would learn **every poly-time function**
- including cryptographic primitives.

Concrete crypto-hardness (realizable PAC, $h^* \in \mathcal{H}$ exists):

- discrete cube root $\to L = O(\log d)$, poly-size threshold networks;
- RSAT $\to L = 2$, $\omega(1)$-width threshold networks;
- lattice shortest-vector $\to$ depth-2, $O(d^{0.001})$-size networks.

Kearns-Valiant 1994; Klivans-Sherstov 2009; Daniely-Reichman 2017.

A compact target network is not enough for efficient learning by any representation.

Why use neural networks anyway?

Compare two universal learners:

Short programs

- universal (every poly-time f);
- poly sample complexity;
- NP-hard proper + crypto-hard improper;
- **discrete, non-differentiable**: no practical local search.

Neural networks

- universal (every poly-time f);
- poly sample complexity;
- NP-hard proper + crypto-hard improper;
- **continuous, differentiable**: gradient descent works in practice.

Equal in expressiveness, sample complexity, and worst-case hardness. NN's edge is practical optimizability.

Where classical theory leaves us

Three questions, three classical answers:

- **Expressive power:** poly-size nets capture time-T computation. **settled**
- **Sample complexity:** VC scales with parameter count E. **loose in practice**
- **Optimization:** worst-case NP/crypto-hard. **gap with practice**

Reframe for the rest of the course:

- **not:** "what about the architecture captures reality?";
- **but:** **"what about real data makes learning tractable?"**

Next: capacity by scale, not count, plus weak learning and boosting.

Summary

Week 5 summary

Week 5: from VC and computation to optimization-driven learning.

Agnostic learning.
- proper $\frac{0}{1}$ ERM can be **NP-hard**
- realizable learning can still be tractable.

Surrogate losses.
- convex losses make linear prediction tractable;
- they **change the comparator**.

Neural networks.
- **expressive:** poly-size nets capture every time-T computation;
- **sample-efficient:** VC scales with parameter count E and depth L;
- **optimization-hard:** NP-hard (Blum-Rivest 1992), crypto-hard improperly.

Next week: capacity by scale, weak learning, boosting.