# Week 10: SGD, Implicit Bias, and Course Summary

## The optimizer is part of the inductive bias

Tianhao Wang
tianhaowang@ucsd.edu

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

# Contents

# Bridge from Week 9

# What Week 9 gave us

Week 9 built online algorithms with regret guarantees: for **any** sequence $z_1, \ldots, z_n$,

$$\frac{1}{n}\sum_{t=1}^{n} \ell(w_t, z_t) \le \inf_{w \in \mathcal{W}} \frac{1}{n}\sum_{t=1}^{n} \ell(w, z_t) + \mathrm{Reg}(n), \quad \mathrm{Reg}(n) = O\left(\frac{BG}{\sqrt{\alpha n}}\right).$$

$B$: comparator-norm bound, $G$: Lipschitz constant of $\ell$, $\alpha$: strong convexity of $\Psi$.

| **FTRL** | **OGD** | **Mirror Descent** |
|---|---|---|
| regularized leader | $\ell_2$ gradient geometry | general $\Psi$ geometry |

Week 9 also gave the **batch counterpart**:

- via stability, the same $\Psi$ makes RERM agnostically PAC learn at rate $O\left(\frac{BG}{\sqrt{\alpha n}}\right)$.

# The one idea this week

Across the course the **inductive bias** has been something we **specify**:

$$\text{class} \;\to\; \text{norm / margin} \;\to\; \text{prior / description} \;\to\; \text{regularizer} \;\to\; \text{regret with } \Psi$$

This week the bias comes from the **optimization algorithm itself**, even with no penalty.

**Arc 1: regret → generalization**

$$\text{low online regret} \Rightarrow \text{low excess risk.}$$

Running the optimizer already learns; **online-to-batch** certifies it for SGD.

**Arc 2: path → which solution**

$$\text{optimization path} \Rightarrow \text{selected minimizer.}$$

Among many fits, the path **selects** one: its implicit bias.

**The optimizer is part of the inductive bias.**

# Online-to-Batch

# Online vs statistical learning

**Arc 1.** One problem $\ell : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}$, comparator class $\mathcal{H}$. Two guarantees for a rule $A$:

**Online regret** holds for **every** sequence $z_1, \ldots, z_n$ (no distribution):

$$\frac{1}{n}\sum_{t=1}^{n} \ell(A(z_1, \ldots, z_{t-1}), z_t) \le \inf_{h \in \mathcal{H}} \frac{1}{n}\sum_{t=1}^{n} \ell(h, z_t) + \mathrm{Reg}(n).$$

**Statistical excess error** holds for **i.i.d.** $S \sim \mathcal{D}^n$ (w.p. $1 - \delta$):

$$L_{\mathcal{D}}(A(S)) \le \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \varepsilon(n, \delta).$$

Same comparator $\inf_{h \in \mathcal{H}}$; they differ only in the **data**: any sequence vs i.i.d.

# Does either guarantee imply the other?

**Batch → online: no**

Low excess error need not give low regret.

Online faces **every** sequence; it cannot use the i.i.d. structure a batch learner exploits.

**Online → batch: yes**

Low regret **does** give low excess error.

The **online-to-batch** conversion turns regret into a generalization bound.

So online regret is the **stronger** guarantee: it implies the statistical one.

That implication, **online-to-batch**, runs the rule on an i.i.d. sample:

$$S = (z_1, \ldots, z_n) \sim \mathcal{D}^n, \quad h_t = A(z_1, \ldots, z_{t-1}), \quad t = 1, \ldots, n.$$

The **stream** $h_1, \ldots, h_n$ must collapse to **one** predictor. Which?

# Regret controls the average, not a single iterate

**Why not the last iterate $h_n$?** Low regret need not control it.

On $\mathcal{H} = [0,1]$ with $\ell(h,z) = h$ (optimum $h^* = 0$), let $A$ **ignore the data**:

$$h_t = 0 \quad \text{for} \quad t < n, \qquad h_n = 1.$$

$$\underbrace{\frac{1}{n}\sum_{t=1}^{n} \ell(h_t, z_t)}_{\frac{1}{n}} - \underbrace{\inf_{h \in \mathcal{H}} \frac{1}{n}\sum_{t=1}^{n} \ell(h, z_t)}_{0} = \frac{1}{n} \quad \text{(average regret)}.$$

Yet $h_n = 1$ is **worst possible**, while the average $1/n$ is **near-optimal**.

A single iterate carries **no guarantee**, so aggregate the trajectory:

- convex loss: **average** the iterates (Jensen's inequality);
- general loss: **randomize**.

# Online-to-batch: the convex case

Run $A$ on the i.i.d. sample, producing iterates $h_t = A(z_1, \ldots, z_{t-1})$.

Return their **average** $\overline{h} = \frac{1}{n}\sum_{t=1}^{n} h_t$.

**Online-to-batch conversion**

Assume $\ell(h, z)$ is **convex** in $h$, and $A$ has regret bound $\mathrm{Reg}(n)$ for **every** sequence. Then

$$\mathbb{E}_{S\sim\mathcal{D}^n} L_{\mathcal{D}}\left(\overline{h}\right) \le \inf_{h\in\mathcal{H}} L_{\mathcal{D}}(h) + \mathrm{Reg}(n).$$

- Left side: the **true risk** of the single predictor $\overline{h}$ we return.
- Right side: the best risk in $\mathcal{H}$, plus the **online regret** as the price.
- Same **excess-risk bound**: the price is now **regret** (was complexity, then stability).

# Proof, step 1: test on a fresh point

$h_t = A(z_1, \ldots, z_{t-1})$ uses only the **past**, so $h_t$ is **independent** of $z_t$.

Condition on $h_t$ and use that $z_t \sim \mathcal{D}$ is fresh:

$$\begin{aligned} \mathbb{E}[\ell(h_t, z_t)] &= \mathbb{E}[\mathbb{E}[\ell(h_t, z_t) \mid h_t]] \quad \text{(tower rule)} \\ &= \mathbb{E}[L_{\mathcal{D}}(h_t)] \qquad\qquad (z_t \text{ fresh, so the inner } \mathbb{E} = L_{\mathcal{D}}(h_t)\,). \end{aligned}$$

Average over $t$:

$$\mathbb{E}\frac{1}{n}\sum_{t=1}^{n} \ell(h_t, z_t) = \mathbb{E}\frac{1}{n}\sum_{t=1}^{n} L_{\mathcal{D}}(h_t).$$

# Proof, step 2: regret, then Jensen

Take expectations in the regret bound; use step 1 on the left:

$$\begin{aligned}
\mathbb{E}\frac{1}{n}\sum_{t=1}^{n} L_{\mathcal{D}}(h_t) &\le \mathbb{E}\inf_h \frac{1}{n}\sum_{t=1}^{n}\ell(h,z_t) + \mathrm{Reg}(n) && \text{(regret + step 1)}\\
&\le \inf_h \mathbb{E}\frac{1}{n}\sum_{t=1}^{n}\ell(h,z_t) + \mathrm{Reg}(n) && (\mathbb{E}\inf \le \inf\mathbb{E})\\
&= \inf_h L_{\mathcal{D}}(h) + \mathrm{Reg}(n) && (\mathbb{E}L_S(h) = L_{\mathcal{D}}(h)).
\end{aligned}$$

Middle step ($\mathbb{E}\inf \le \inf\mathbb{E}$): for any fixed $h'$, $\mathbb{E}\inf_h L_S(h) \le \mathbb{E}L_S(h') = L_{\mathcal{D}}(h')$; take inf over $h'$.

Finally, **convexity** of $L_{\mathcal{D}}$ + Jensen's inequality

$$\mathbb{E}L_{\mathcal{D}}\left(\overline{h}\right) = \mathbb{E}L_{\mathcal{D}}\left(\frac{1}{n}\sum_{t=1}^{n} h_t\right) \le \mathbb{E}\frac{1}{n}\sum_{t=1}^{n} L_{\mathcal{D}}(h_t) \le \inf_h L_{\mathcal{D}}(h) + \mathrm{Reg}(n)$$

# The general case: randomize

Averaging worked because the loss was convex. It **breaks down** when:

- the $h_t$ are **classifiers**: their average is real-valued, not a classifier;
- $L_{\mathcal{D}}$ is **non-convex**: Jensen fails, so $L_{\mathcal{D}}\left(\overline{h}\right)$ may exceed the average risk.

Instead, return a **randomized** predictor:

$$\text{draw} \quad T \sim \text{Uniform}(\{1, \ldots, n\}), \quad \text{predict with} \quad h_T.$$

By **linearity** (no convexity), its expected risk **equals** the trajectory average:

$$\mathbb{E}_T L_{\mathcal{D}}(h_T) = \frac{1}{n} \sum_{t=1}^{n} L_{\mathcal{D}}(h_t) \leq \inf_h L_{\mathcal{D}}(h) + \text{Reg}(n),$$

where the last bound is step 2.

> Convexity averages **predictions** (Jensen, $\leq$); randomization averages **losses** (linearity, $=$).

# The same $1/\sqrt{n}$ rate, now from regret

Specialize: $w$ with $\|w\|_2 \le B$ (Week 9), $\Psi = \frac{1}{2}\|\cdot\|_2^2$ (so $\alpha = 1$), $\mathrm{Reg}(n) = O\left(\frac{BG}{\sqrt{n}}\right)$:

$$\mathbb{E}L_{\mathcal{D}}(\overline{w}) \le \inf_{\|w\|_2 \le B} L_{\mathcal{D}}(w) + O\left(\frac{BG}{\sqrt{n}}\right).$$

Invert for excess error $\varepsilon$:

$$n = O\left(\frac{B^2G^2}{\varepsilon^2}\right).$$

Three routes to the **same $1/\sqrt{n}$ rate**:

- Rademacher (Week 7);
- RERM stability (Week 8);
- online-to-batch (today).

- First two routes: bound a **minimizer** of $L_S$, plus a convergence or stability step.
- Online-to-batch: bound the **algorithm's output**, from regret alone.

# Example: online-to-batch for least squares

- Squared loss $\ell(w, (x, y)) = (\langle w, x\rangle - y)^2$ is **convex** in $w$.
- Assume $\|x\|_2 \le R$, $|y| \le Y$, comparator $\mathcal{W} = \{\|w\|_2 \le B\}$.

OGD on the i.i.d. stream takes the gradient step

$$w_{t+1} = \Pi_{\mathcal{W}}(w_t - \eta_t \cdot 2(\langle w_t, x_t\rangle - y_t)x_t).$$

Its loss is $G$-Lipschitz on $\mathcal{W}$, $G = 2R(BR + Y)$, so the **averaged predictor** $\overline{w}$ satisfies

$$\mathbb{E}L_{\mathcal{D}}(\overline{w}) \le \inf_{\|w\|_2 \le B} L_{\mathcal{D}}(w) + O\left(\frac{BG}{\sqrt{n}}\right).$$

- Usual least squares: solve the normal equations over the whole sample at once.
- Here: **one gradient step per point**, a single pass, then average.

# SGD as a Learning Rule

# Two readings of the same update

Week 9′s OGD, the rule we just ran on least squares, is a **gradient step**:

$$w_{t+1} = w_t - \eta_t g_t.$$

On the i.i.d. data stream, that same update is what optimization calls **SGD**.

| Setting | Data | Goal |
|---|---|---|
| Online learning | $z_t$ may be adversarial | low regret on the observed sequence |
| Stochastic approximation | $z_t \sim \mathcal{D}$ i.i.d. | minimize population risk $L_{\mathcal{D}}(w)$ |

- Online learning is where the **guarantee** lives: Week 9′s regret bound.
- SGD is the **method** we run on i.i.d. data.
- Same update, different question.

**Week 9′s regret bound certifies SGD, via online-to-batch.**

# The SGD update

Objective: minimize the **population risk**

$$\min_w L_{\mathcal{D}}(w), \quad L_{\mathcal{D}}(w) = \mathbb{E}_{z\sim\mathcal{D}}\ell(w, z).$$

At step $t$, draw a **fresh** $z_t \sim \mathcal{D}$ and take the gradient

$$g_t = \nabla_w \ell(w_t, z_t), \quad w_{t+1} = w_t - \eta_t g_t.$$

Return the **averaged iterate** $\overline{w}_n = \frac{1}{n}\sum_{t=1}^{n} w_t$.

- One **fresh** example per step, so $n = T$, and no constraint to project onto.
- Step size $\eta_t = \Theta\big(B/\big(G\sqrt{t}\big)\big)$ matches OGD; the bound competes with the best $\|w\|_2 \le B$.

# SGD, the algorithm

**Stochastic Gradient Descent (fresh sample)**

**Input:** step sizes $\eta_1, \dots, \eta_n$.

**Init:** $w_1 = 0$.

**For** $t = 1, \dots, n$:

- draw a **fresh** $z_t \sim \mathcal{D}$;
- gradient $g_t = \nabla_w \ell(w_t, z_t)$;
- step $w_{t+1} = w_t - \eta_t g_t$.

**Return** $\overline{w}_n = \frac{1}{n} \sum_{t=1}^{n} w_t$.

- One **gradient per step**, one example at a time: cost $O(d)$ per step, not $O(nd)$.
- The only change from full-batch GD: the gradient uses **one sample**, not all $n$.

# Why the noisy gradient is legitimate

Here $g_t = \nabla \ell(w_t, z_t)$. Since $z_t$ is fresh and independent of $w_t$,

$$\mathbb{E}[g_t \mid w_t] = \mathbb{E}_{z \sim \mathcal{D}} \nabla \ell(w_t, z) = \nabla L_{\mathcal{D}}(w_t).$$

So $g_t$ is an **unbiased** estimate of the population gradient.

- SGD descends $L_{\mathcal{D}}$ **in expectation**, using one example to estimate the direction.
- The gradient noise is exactly what the $1/\sqrt{n}$ rate pays for.

# Example: SGD for logistic regression

Logistic loss $\ell(w, (x, y)) = \log\big(1 + e^{-y\langle w, x\rangle}\big)$ with $y \in \{\pm 1\}$.

Its gradient uses the **sigmoid** $\sigma(u) = 1/(1 + e^{-u})$:

$$\nabla_w \ell(w, (x, y)) = -yx \cdot \sigma(-y\langle w, x\rangle).$$

The SGD step on a fresh $(x_t, y_t)$ is therefore

$$w_{t+1} = w_t + \eta_t y_t x_t \cdot \sigma(-y_t \langle w_t, x_t\rangle).$$

- The factor $\sigma(-y_t\langle w_t, x_t\rangle)$ is the model's **probability of error** on $(x_t, y_t)$.
- **Confident-correct** points barely move $w$.
- **Misclassified** points move it most.

# SGD learns, directly

**Corollary of online-to-batch**

If $\ell(\cdot, z)$ is convex and $G$-Lipschitz w.r.t. $\|\cdot\|_2$, then fresh-sample SGD with averaging satisfies

$$\mathbb{E} L_{\mathcal{D}}(\overline{w}_n) \leq \inf_{\|w\|_2 \leq B} L_{\mathcal{D}}(w) + O\left(\frac{BG}{\sqrt{n}}\right).$$

The **same $O(BG/\sqrt{n})$ bound** as before, now read as an algorithm guarantee.

- Online-to-batch on OGD: regret, i.i.d. independence, averaging Jensen.

# Fresh samples vs reusing a training set

- Fresh-sample SGD drew a new $z_t \sim \mathcal{D}$ each step.
- In practice we **reuse** a finite $S$, resampling an index $i \sim \{1, \ldots, n\}$:

| Mode | Objective | Update step | Norm control |
|---|---|---|---|
| Fresh-sample | $\min_w L_{\mathcal{D}}(w)$ | $w_{t+1} = w_t - \eta_t \nabla \ell(w_t, z_t)$, fresh $z_t$ | none |
| ERM on $S$ | $\min_{\|w\|_2 \le B} L_S(w)$ | $w_{t+1} = \Pi_{\mathcal{W}}(w_t - \eta_t \nabla \ell(w_t, z_i))$ | projection |
| RERM on $S$ | $\min_w L_S(w) + \frac{\lambda}{2}\|w\|_2^2$ | $w_{t+1} = w_t - \eta_t(\nabla \ell(w_t, z_i) + \lambda w_t)$ | shrinkage |

Reusing $S$ still learns. Next: guarantees for **SGD on ERM** and **SGD on RERM**.

# SGD on a fixed sample: the guarantee

**SGD on ERM**: minimize $L_S$ over $\|w\|_2 \le B$ (projection), resampling indices for $T$ steps.

Excess risk over the best $\|w\|_2 \le B$ is bounded by optimization plus estimation:

$$L_{\mathcal{D}}(\overline{w}_T) - \inf_{\|w\|_2 \le B} L_{\mathcal{D}}(w) \le \underbrace{O\left(\frac{BG}{\sqrt{T}}\right)}_{\text{optimization}} + \underbrace{O\left(\frac{BG}{\sqrt{n}}\right)}_{\text{estimation}}.$$

- **Optimization** $\to 0$ as $T$ grows: SGD minimizes $L_S$.
- **Estimation** $O((BG)/\sqrt{n})$: norm control bounds the gap $L_{\mathcal{D}} - L_S$ (Week 8).
- **SGD on RERM** (shrinkage, not projection): same rate, the Week 8 stability bound.

**Same $O(BG/\sqrt{n})$ rate as fresh-sample SGD, but here the norm control does the regularizing.**

# Three sources of excess risk

Optimizing $L_S$ on a fixed sample, the excess risk has **three** sources, one term each:

$$L_{\mathcal{D}}(w_T) - \inf_h L_{\mathcal{D}}(h) = \underbrace{[L_{\mathcal{D}}(w_T) - L_{\mathcal{D}}(\hat{w})]}_{\text{optimization}} + \underbrace{[L_{\mathcal{D}}(\hat{w}) - L_{\mathcal{D}}(w^*)]}_{\text{estimation}} + \underbrace{\left[L_{\mathcal{D}}(w^*) - \inf_h L_{\mathcal{D}}(h)\right]}_{\text{approximation}}.$$

$\hat{w}$: ERM minimizer of $L_S$; $w^*$: best in class; $\inf_h L_{\mathcal{D}}$: best possible.

- **Estimation** is the generalization gap; we controlled it through the **norm**.
- But plain GD on $L_S$ writes **no penalty**. So where is its norm control?

# Implicit Bias

# Where is the regularization?

GD on $L_S$ writes **no penalty**, so what decides which solution it reaches?

Its gradient is $g_t = \nabla L_S(w_t)$. **One step** solves a **proximal** problem,

$$w_{t+1} = \operatorname{argmin}_w \langle g_t, w \rangle + \frac{1}{2\eta_t} \|w - w_t\|_2^2.$$

Setting the gradient to zero, $g_t + \frac{1}{\eta_t}(w_{t+1} - w_t) = 0$, i.e. $w_{t+1} = w_t - \eta_t g_t$.

Read it as **(linear model of the loss)** plus **(stay near $w_t$)**: a per-step $\|\cdot\|_2$ penalty.

Mirror descent swaps it for a Bregman divergence (Week 9):

$$w_{t+1} = \operatorname{argmin}_w \langle g_t, w \rangle + \frac{1}{\eta_t} D_\Psi(w \,\|\, w_t).$$

# Which minimizer does the optimizer pick?

Classical: fix **what** we minimize over.

Which hypothesis class, norm, or regularizer?

Modern: with high capacity, **many** $w$ achieve $L_S(w) = 0$.

The class and the loss **cannot tell them apart**: all fit. So ask:

Among all minimizers, which one does the optimizer select?

Nothing we **wrote down** (class, loss, penalty) breaks this tie.

The only remaining tiebreaker is the **algorithm** that produced $w$.

**That selection is the optimizer's implicit bias: Arc 2.**

# Many minimizers, different fates

In overparameterized problems ($w \in \mathbb{R}^d$ with $d > n$) the zero-loss set

$$\{w : L_S(w) = 0\}$$

is large. For linear interpolation it is an **affine subspace** of dimension $\geq d - n$:

- the $n$ constraints $\langle w, x_i \rangle = y_i$ fix $n$ directions;
- the remaining $d - n$ directions are free.
- All these $w$ have the **same** training loss 0.
- Their **true risks differ**: they agree on the data, but disagree on **new** inputs.

> Optimization here is not "find a minimizer," it is "**choose** a minimizer."

# Gradient descent stays in the data span

Underdetermined least squares, $n < d$, with a **feature map** $\varphi$ (linear case $\varphi(x) = x$), interpolation feasible:

$$y_i = \langle w, \varphi(x_i) \rangle \quad \text{for all } i.$$

The gradient of the **squared loss** $L_S(w) = \frac{1}{n} \sum_{i=1}^{n} \left( \langle w, \varphi(x_i) \rangle - y_i \right)^2$ is a combination of the features:

$$\nabla L_S(w) = \frac{2}{n} \sum_{i=1}^{n} (\langle w, \varphi(x_i) \rangle - y_i) \varphi(x_i) \in \operatorname{span}\{\varphi(x_1), ..., \varphi(x_n)\}.$$

Starting at $w_0 = 0$, every step adds a vector from the span, so

$$w_t \in V := \operatorname{span}\{\varphi(x_1), ..., \varphi(x_n)\} \quad \text{for all } t.$$

# GD selects the minimum-norm interpolator

Any interpolator splits as $w = w_{\parallel} + v_{\perp}$ with $w_{\parallel} \in V$ and $v_{\perp} \in V^{\perp}$.

- The prediction $\langle w, \varphi(x_i) \rangle$ depends only on $w_{\parallel}$, since $\varphi(x_i) \in V$; so $v_{\perp}$ is **invisible** to the data.
- GD lives in $V$, so it leaves $v_{\perp} = 0$: among all interpolators it returns the one with **no $V^{\perp}$ part**.

GD on this convex least-squares objective converges to an interpolator; from $w_0 = 0$ that is

$$\operatorname{argmin}_w \|w\|_2 \quad \text{s.t.} \quad \langle w, \varphi(x_i) \rangle = y_i \quad \text{for all } i.$$

This is **Week 8's $\|\cdot\|_2^2$ RERM with $\lambda \to 0$**, reached with **no penalty**.

# Why the min-norm interpolator generalizes

Selecting an interpolator helps only if the selected one **generalizes**.

Recall Week 7: for linear predictors with $\|w\|_2 \le B$ and features $\|\varphi(x)\|_2 \le R$,

$$L_{\mathcal{D}}(w) \le L_S(w) + O\left(\frac{GBR}{\sqrt{n}}\right).$$

Every interpolator has $L_S(w) = 0$, so the bound is driven by the **norm** $B = \|w\|_2$.

GD picks the smallest-norm one,

$$w_{\mathrm{GD}} \in \operatorname{argmin}_{L_S(w)=0} \|w\|_2,$$

which therefore has the **tightest** form of this bound.

**GD's implicit bias selects the interpolator with the best norm-based guarantee.**

# Different algorithm, different norm: Boosting

Recall Week 6: AdaBoost is **coordinate descent** on the exponential loss. The ensemble

$$f_T(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

is a **linear predictor** over weak-rule features $\varphi(x)_h = h(x)$, with weight vector

$$w = \sum_{t=1}^{T} \alpha_t e_{h_t}, \quad \|w\|_1 = \sum_{t=1}^{T} \alpha_t.$$

- Coordinate descent moves **one weight at a time**.
- Its implicit bias is the $\ell_1$ **margin** (Week 6), not GD's $\ell_2$ minimizer.

**Change the algorithm, change the norm, change the bias.**

# Separable data: the minimizer runs to infinity

Earlier cases: **finite interpolators**. Separable data: **no finite minimizer** at all.

Logistic loss with labels $y_i \in \{\pm 1\}$:

$$L_S(w) = \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y_i \langle w, x_i \rangle}\right).$$

If **separable** (some $w$ has all $y_i \langle w, x_i \rangle > 0$), then $cw$ with $c \to \infty$ sends each term to 0:

$$\inf_w L_S(w) = 0, \quad \text{but no finite minimizer.}$$

Every margin $y_i \langle w, x_i \rangle$ must grow unboundedly, so GD drives $\|w_t\|_2 \to \infty$, $L_S(w_t) \to 0$.

> $w_t$ itself diverges; what converges is its **direction** $w_t / \|w_t\|_2$.

# GD finds the max-margin direction

For separable logistic regression, GD's direction converges to the **hard-margin** solution:

$$\frac{w_t}{\|w_t\|_2} \to \frac{\hat{w}}{\|\hat{w}\|_2}, \quad \hat{w} = \operatorname{argmin}_w \|w\|_2 \ \text{ s.t. } \ y_i \langle w, x_i \rangle \geq 1 \ \text{ for all } i.$$

- $\hat{w}$ is exactly the **max-margin / SVM** predictor of Week 7 (margin $= 1/\|\hat{w}\|_2$).
- There is **no margin term** in the logistic objective; GD supplies it implicitly.
- Convergence in direction is **slow** (logarithmic in $t$).

**The optimizer performs, for free, the norm minimization Week 7 imposed by hand.**

# Summary: optimizer to bias

| Method | Geometry | Selected solution | Explicit twin |
|---|---|---|---|
| GD from 0 | $\ell_2$ | min Euclidean norm | Week 8 $\|\cdot\|_2^2$ |
| Coord. descent / Boosting | $\ell_1$ | max $\ell_1$ margin | Week 6 $\ell_1$ margin |
| Mirror descent | $\Psi$ / $D_\Psi$ | min $\Psi$ from $w_0$ | Week 9 $\Psi$ |
| GD, separable logistic | $\ell_2$ direction | max-margin separator | Week 7 margin |

**Each implicit bias matches an explicit regularizer from earlier in the course.**

# Neural Networks Revisited

# Back to neural networks

Week 5 raised three questions about neural networks:

| | |
|---|---|
| Expressive power | can the architecture represent good functions? |
| Sample complexity | how many samples to generalize? |
| Computation | why does local search find a useful solution? |

Implicit bias **reframes** the third question:

not "does training reach *a* global minimum?" but "**which** one does it reach?"

- The architecture admits many zero-train-error solutions, with **different test error**.
- Training selects one: that is its implicit bias.

# Parameter count is not enough

Overparameterized nets fit **many** functions, including ones that do not generalize.

So capacity measured by parameter count is **vacuous** here. Two separate questions remain:

Does the architecture contain a good predictor?

Which predictor does training select?

The selected predictor can depend on:

| | | |
|---|---|---|
| architecture | initialization | step size |
| batch size | optimizer | training time |

Empirical, not a theorem: generalization tracks the optimizer's **implicit bias**.

# Evidence: fitting random labels

Zhang et al. (2017): **randomize the labels** on a standard overparameterized network.

- The network still reaches **zero training error**: it can fit pure noise.
- So its capacity is large enough to memorize **any** labeling of the data.
- On the **true** labels, the same architecture and optimizer generalize well.

> - A bound that permits fitting random labels cannot explain real-label generalization.
> - The difference comes from the **data and the optimizer**, not the architecture's size.

# Double descent

Classical theory: test risk **rises** once a model can overfit.

Overparameterized models show a **second descent**:

| Model size | Regime | Test risk |
|---|---|---|
| params $<$ data ($< n$) | classical | U-shaped: under- then over-fit |
| params $\approx$ data ($\approx n$) | interpolation threshold | risk **peaks** |
| params $>$ data ($> n$) | overparameterized | risk **descends again** |

- Past the threshold, the optimizer's **implicit bias** selects a low-norm interpolant.
- A monotone capacity bound predicts the wrong shape.

# Implicit bias in deep learning

The linear results have **partial** analogues for nonlinear networks (active research):

- **Homogeneous** networks, separable data: GD finds the **max-margin** direction.
- **Gradient flow** on deep linear networks favors **low-rank** solutions.
- SGD is observed to prefer **flat** minima, which are argued to generalize better.

These cases are partly understood; the general nonlinear case remains **open**.

# Adaptive optimizers change the geometry

Week 9 fixed one geometry $\Psi$. Modern optimizers **adapt** it using past gradients.

| | |
|---|---|
| AdaGrad | scale coordinates by accumulated squared gradients |
| Adam | moving averages of gradients and squared gradients |
| Shampoo | matrix-shaped preconditioners |

- A time-varying $\Psi$ means a **time-varying implicit bias**.
- Which solution adaptive methods prefer is active research.

**That is the research frontier. Step back: the whole course on one map.**

# Course Summary

# The 10-week map

| Week | Topic | Core question |
|---|---|---|
| 1 | Online learning and No Free Lunch | what assumptions make learning possible? |
| 2 | Statistical learning and VC theory | when does ERM generalize? |
| 3 | PAC learning and Fundamental Theorem | what exactly is learnable? |
| 4 | Non-uniform learning and computation | how do prior, description length, and efficiency matter? |
| 5 | Hardness, surrogates, neural networks | what can tractable algorithms hope to learn? |
| 6 | Boosting and compression | how do weak rules and small descriptions generalize? |
| 7 | Real-valued and scale-sensitive complexity | how do norm and margin replace dimension? |
| 8 | Stability and RERM | why do stable rules generalize? |
| 9 | Online regret and Mirror Descent | how does geometry control sequential learning? |
| 10 | SGD and implicit bias | which solution does optimization choose? |

# Generalization mechanisms, explicit to implicit

| Bias source | Quantity controlled | Analysis tool |
|---|---|---|
| finite class | $\log\lvert\mathcal{H}\rvert$ | union bound / Halving |
| VC theory | $\mathrm{VCdim}(\mathcal{H})$ | growth function / Sauer |
| prior / description | description length / prior | MDL / SRM / PAC-Bayes |
| compression | number of stored examples | sample compression bound |
| norm / margin | norm, margin, Rademacher | contraction / fat-shattering |
| stability / RERM | sensitivity of $A(S)$ | RERM stability |
| online regret | distance to best fixed comparator | FTRL / OGD / MD |
| implicit bias | solution selected by optimizer | GD / SGD path analysis |