Experiments to show implicit bias of Gradient Descent (GD)

The simplest is to start with understanding the implicit biases of GD. One implicit bias of GD is that it is biased toward low rank solutions. This has been shown in deep linear networks by Saxe et al. 2014 studying the exact solutions of gradient flow in deep linear networks. Main implicit biases (see deep research):

- Low rank solutions and first learns largest eigenvalues. For ex see paper.
- Avoiding sharp minima thanks to edge of stability.
- Feature averaging; here
- Spectral bias: learning low frequency features first paper
- Directional convergence of normalized weights and alignment of gradient with weights. Also implicit bias toward max margin solutions paper
- Conjecture: the gradient updates are themselves low rank?

Most interesting to look into are: low rank solutions, learning more important features first, edge of stability and checking GD vs Gradient flow (continuous time limit of GD). Another important paper is the saddle-to-saddle dynamics of GD by Jacot et al.

Experimental Protocol: Implicit Bias Toward Low-Rank Solutions

Overview. This protocol empirically demonstrates that gradient-based optimization favours low-rank solutions when fitting a low-rank target matrix. Both discrete-time *gradient descent* (GD) and its continuous-time limit *gradient flow* (GF) are tested across network depths, initializations, and learning rates.

Network Architecture

Teacher (Target) Network.

- Dimension: $n = 100$ with square matrices. If dimension is too large, use smaller matrices to run quicker experiments, can also use rectangular matrices. The important part is having a deep student network in the overparameterized regime.
- Rank, for example: $r \in \{5, 10, 20\}$.
- Generate Teacher network of Depth N

Student (Overparameterized) Network.

- Deep *linear* network: $W_N \cdots W_1$
- Depth $N \geq 2$; the model is overparameterized for any $N \geq 2$.

Step 1: Orthonormal bases. Generate two random orthonormal matrices

$$U, V \in \mathbb{R}^{100 \times r}.$$

Only their first r columns matter, so obtain them efficiently via a *thin* QR decomposition applied to two independent $100 \times r$ standard-normal matrices.

Step 2: Singular values. Create a length-r vector $\boldsymbol{\sigma}$, for example, with exponentially decaying entries (but can make other choices)

$$\sigma_i = 10 \cdot 0.5^{\,i-1}, \qquad i = 1, \ldots, r,$$

ensuring clear separation of spectral modes.

Step 3: Assemble the matrix. Define

$$W_{\text{teacher}} \;=\; U\,\mathrm{diag}(\boldsymbol{\sigma})\,V^{\top}.$$

Because both U and V have orthonormal columns of length r, the product has

$$\mathrm{rank}\big(W_{\text{teacher}}\big) \;=\; r.$$

Initialization (Saddle–to–Saddle Regime)

1. **Gaussian:** $W_i \sim \mathcal{N}\big(0, \sigma^2\big)$ with $\sigma = w^{-\gamma}$ with w the width of the given layer and $\gamma > 1$ (see saddle-to-saddle dynamics by Jacot et al. for more details).

Training Setup

Mean square error Loss.

$$\mathcal{L} = \tfrac{1}{2}\|W_N \cdots W_1 - W_{\text{teacher}}\|_F^2.$$

Optimizers.

- **Gradient Descent (GD):** $W_i^{(t+1)} = W_i^{(t)} - \eta\, \frac{\partial \mathcal{L}}{\partial W_i}$, $\eta \in \{10^{-4}, 10^{-3}, 10^{-2}\}$.
- **Gradient Flow (GF):** $\Delta W_i t = -dt\, \frac{\partial \mathcal{L}}{\partial W_i}$; approximate with GD using smaller dt than η. Use Runge-Kutta 4th order method to approximate the flow.

Metrics

Rank-related quantities. Let $\|W\|_*$ be the nuclear norm of W (sum of singular values) and $\|W\|_F$ be the Frobenius norm. In addition to tracking rank via non zero singular values, we also track the effective rank, the stable rank. The latter grow more smoothly and show the incremental dynamics of the rank a bit better.

$$r_{\text{eff}}(W) = \frac{\|W\|_*}{\|W\|}, \qquad r_{\text{stable}}(W) = \frac{\|W\|_F^2}{\|W\|^2}, \qquad r_{\text{num}}(W) = \#\{\sigma_i > 10^{-6}\}.$$

Tracked every iteration.

- Layer ranks $r(W_i)$, product rank $r(W_N \cdots W_1)$.
- Singular values of $W_N \cdots W_1$.
- Training loss $\mathcal{L}$.
- Rank of the updated weights (ΔW_i) over time.
- Alignment of learned vs. teacher singular vectors (i.e. their angles, scalar product)

Experimental Procedure

1. Initialize network at saddle-to-saddle regime.
2. For example, train for 10 000–50 000 iterations (until convergence). **Important: also use iterations instead of epochs to show the incremental dynamics.**
3. Log metrics every 10–50 iterations; save checkpoints every 1000 iterations.

GD vs. GF comparison. Use identical initializations; for GF pick a small step size and plot against continuous time $t = \eta \times$ iteration. Compare rank dynamics and loss convergence for both. Run the comparison for different learning rates across different order of magnitude.

Parameter sweeps. Keep initialization fixed. These are examples of parameters to sweep:

$$r \in \{5, 10, 20\},\; N \in \{2, 3, 4, 5\},\; \eta \in \{10^{-4}, 10^{-3}, 10^{-2}\}.$$

Visualizations

1. **Effective rank evolution:** $r_{\text{eff}}(W_N \cdots W_1)$ vs. iteration/t.
2. **Singular value dynamics:** log–linear plot of all $\sigma_i(t)$.
3. **Layer-wise rank:** subplots of each $r(W_i)$ over time.
4. Also plot rank of the updated weights (ΔW_i) over time.
5. **Loss convergence:** $\mathcal{L}$ vs. iteration (log scale).

Expected Results

- Discrete rank "plateaus" at 1,2,...,r.
- Larger singular values converge first (bias toward important features first).
- GF shows smoother transitions than GD.
- Deeper networks exhibit stronger low-rank bias.

Implementation Notes

Reproducibility. Fix random seeds, log hyperparameters, and version-control code.

Optional Extensions

A. Lazy (NTK) regime:
B. Rectangular matrices: $W_{\text{teacher}} \in \mathbb{R}^{m \times n}$; track singular values.