15) Weight Inits

Weight Matrix

- Given $y = xW^T$, where $W = \begin{bmatrix} - & w_1^T & - \\ - & w_2^T & - \\ & \vdots & \\ - & w_n^T & - \end{bmatrix}$ → each row w_i = weights for node i
  # column = # nodes in this layer
  → so W.shape = (output, input)

  y ↑ output layer, x ↑ input layer, W ↑ weight matrix

- Visual: batch=5

$$\underset{\text{input}}{\begin{bmatrix} x_0 \\ \vdots \\ x_{10} \end{bmatrix}} \xrightarrow[W^T:(10,14)]{X:(5,10)} \underset{\text{hidden 1}}{\begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_{14} \end{bmatrix}} \xrightarrow[W^T:(14,19)]{X:(5,14)} \underset{\text{hidden 2}}{\begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_{19} \end{bmatrix}} \xrightarrow[W^T:(19,8)]{X:(5,19)} \underset{\text{output}}{\begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_8 \end{bmatrix}} \xrightarrow[W^T:(8,8)]{X:(5,8)} \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_8 \end{bmatrix}^T$$

size: (5,8)

Weight Initialization

- Weight Symmetry: issue where model has same params (~0 gradients)
- Solution: Random Weights provides model the variability & direction to learn
  so draw weight values from ① Normal $\mathcal{N}$ ② Uniform $\mathcal{U}$
- Weight-Init Algorithms:

① Kaming: draw $w \in \mathcal{U}(-\sigma, \sigma)$ with $\sigma^2 = \frac{1+a^2}{N_{in}}$ ← slope of activation (negative part) = 1 for linear, 0 for ReLU

② Xaiver: draw $w \in \mathcal{N}(0, \sigma^2)$ with $\sigma^2 = \frac{2}{N_{in}+N_{out}}$ ← # input & output features

Freezing Weights

- Deactivate gradient descent in a layer (for transfer learning)
- How?: set requries_grad = False for that layer

Quantify Weight Changes

- Weight matrix changes over time (its distribution widens)
- metrics:

① Frobenius norm $d = \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{n} \left(W_{i,j}^{(t)} - W_{i,j}^{(t-1)}\right)^2}$ (t-th epoch) } large d = large change in weights = a lot of learning

② Condition number $K = \sigma_{max} / \sigma_{min}$ (set of SVD #) } large K = sparse weight matrix = layer learned specific features