CS 182 Lecture 7: Initialization, BatchNorm 6/15

$\frac{df}{dW^{(1)}} = \frac{dz^{(1)}}{dW^{(1)}} \frac{df}{dz^{(1)}} = \delta x^T$ ← if x is "imbalanced", so are gradients.
Therefore, we really want all entries in x → same scale.

standardization: $\mu = 0$, $\sigma = 1$.

$\bar{x}_i = x_i - E[x]$ $\quad \bar{x}_i = \frac{x_i - E[x]}{\sqrt{E[(x_i - E[x])^2]}}$ ← $\bar{\sigma}_i$ $\quad E[x] \approx \frac{1}{N} \sum_{i=1}^{N} x_i$

Basic idea of Batch Norm: standardize activations at each layer, controlling gradients.
computationally expensive idea, so only perform in batch.

$\mu^{(1)} \approx \frac{1}{B} \sum_{j=1}^{B} a_{ij}^{(1)}$ $\quad \sigma^{(1)} \approx \sqrt{\frac{1}{B} \sum_{j=1}^{B} (a_{ij}^{(1)} - \mu^{(1)})^2}$ $\quad \bar{a}_i^{(1)} = \frac{a_i^{(1)} - \mu^{(1)}}{\sigma^{(1)}} \gamma + \beta$ ← learnable scale and bias same dim as $\bar{a}_i^{(1)}$

→ can be trained via backprop since these are differentiable.
→ can be placed either before/after nonlinearity.
→ we can often use a larger learning rate
→ models can train faster
→ generally requires less regularization

Basic initialization methods: ensure activations are on reasonable scale that stays roughly constant
More advanced init. methods involve eigenvalues/Jacobians
"Try" to have well-behaved gradients from the get-go

set $W_{ij} \sim N(0, \sigma^2_W)$, $b_i \approx 0$, $\quad z_i = \sum_j W_{ij} a_j + \cancel{b_i}$ $\quad$ assume $a_j \sim N(0, \sigma_a)$

so, set std of $W_{ij} = 1/\sqrt{D_a}$ $\quad E[z_i^2] = \sum_j E[W_{ij}^2] E[a_j^2] = D_a \sigma^2_W \sigma^2_a$ ← dim of a

"Xavier initialization" If $D_a \sigma^2_W > 1$, magnitude grows w/ each layer
ReLU issue: zeros out activations. If $D_a \sigma^2_W < 1$, magnitude shrinks w/ each layer → $\sigma^2_W = \frac{1}{D_a}$
Ergo, increase std of W_{ij} → $1/\sqrt{\frac{1}{2} D_a}$ (proposed in ResNet).
Alternative init. $b_i = 0.1$ (or small constant) to avoid any zero-out in ReLU.

Reminder: $\frac{df}{dW^{(1)}} = J_1 J_2 J_3 \cdots J_n \frac{df}{dz^{(n)}}$ $\quad \forall J_i, \; J_i = U_i \Lambda_i V_i$ (SVD). ← diag. matrix of λ's of J_i.
By ensuring Λ_i's are scaled, we avoid 0/infinity convergence issue.
Since $J = W^T$ (derivative is W) → $W^{(i)} = U^{(i)} \Lambda^{(i)} V^{(i)}$, force $\Lambda^{(i)} = I$, $W^{(i)} \leftarrow U^{(i)} V^{(i)}$.

"Measure of last resort": Gradient clipping, because "monster gradients" can occur
→ per-element clipping: $\bar{g}_i \leftarrow \max(\min(g_i, c_i), -c_i)$
$-c_i < g_i < c_i$
→ norm clipping:
$\bar{g}_i \leftarrow g \frac{\min(\|g\|, c)}{\|g\|}$ ← clips length, not dimension; choose c_i thru experimentation ← see what "healthy" magnitudes look like (plot).

NNs → high variance (lots of parameters), but with multiple, more agreement on "right"
Variance $= E_{D \sim p(D)}[\|f_D(x) - \bar{f}(x)\|^2]$ $\quad \bar{f}(x) = E_{D \sim p(D)}[f_D(x)] \approx \frac{1}{M} \sum_{i=1}^{M} f_{D_i}(x)$ ← bootstrap sample

Select? → principled approach: average ($p(y|x) = \frac{1}{M} \sum_{j=1}^{M} p_{\theta_j}(y|x)$, often more robust)
→ simple approach: majority vote (analogous to "First Past The Post")
Ensembling in practice: train M models $p_{\theta_j}(y|x)$ on the same training set, maj. vote.
Even faster ensembles? Share features for all M models. Separate ensemble of linear classifier "heads" at end (these are task-specific)
Snapshot ensembles: save our parameters as "snapshots" during SGD, use later as a model.
The bigger the ensemble, the better, the more expensive.
Dropout: randomly set some activations to 0 in forward pass. Creates a "new" network, helps w/ ensembling out of a single NN. 2^{km} diff. models (k layers, m nodes/layer)
At test time: $\bar{W}^{(i)} = \frac{1}{2} W^{(i)}$, since on average, $\frac{1}{2}$ of dimensions are forced to 0.
Hyperparameters that affect generalization (validation): ensembling, dropout, arch.
General method to pick hyperparameters, coarse-to-fine, broad sweep before "zeroing in"
↳ In practice: random hp. search > grid search.