Advanced techniques

Lecture 5

Changho Suh

September 28, 2021

Weight initialization & techniques for training stability

Outline

1. Weight initialization

   Xavier's initialization

   He's initialization

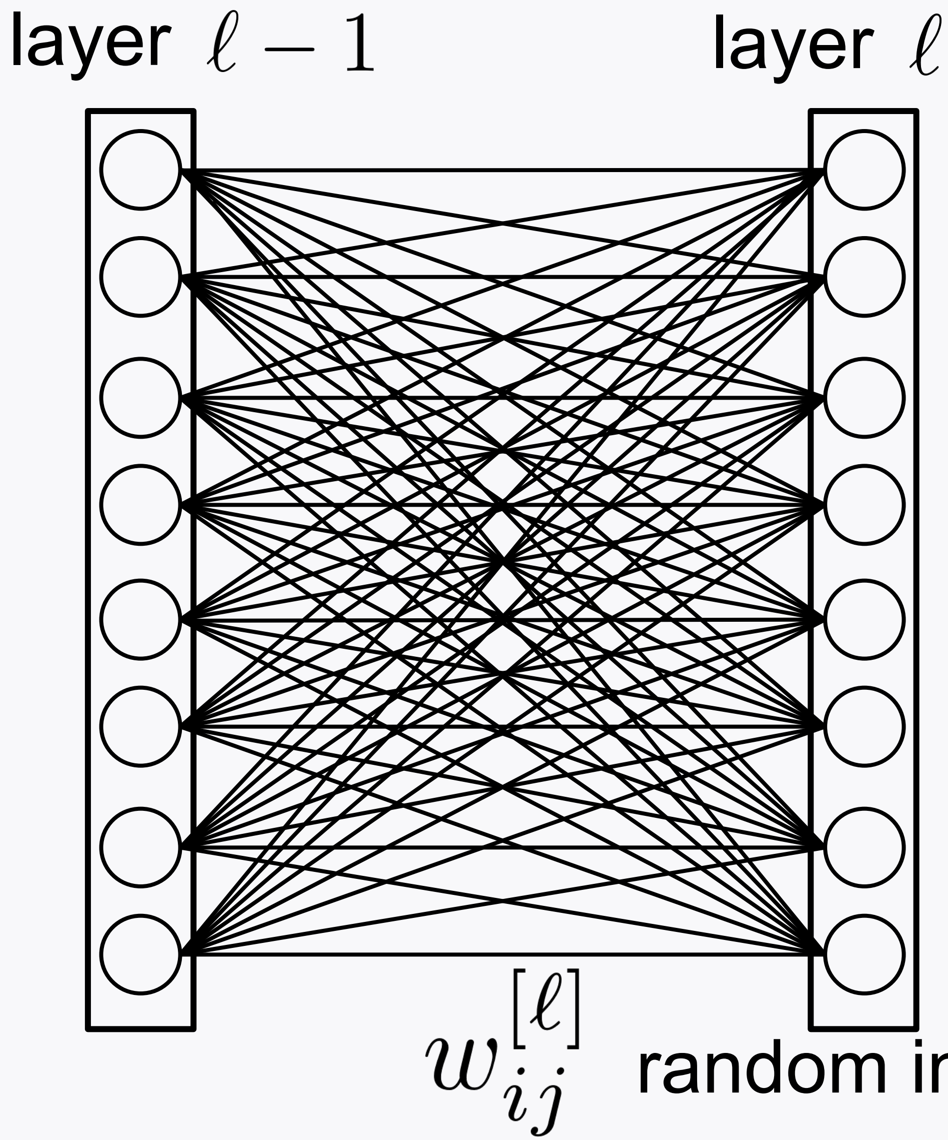
2. Techniques for training stability

   Adam optimizer

   Learning rate decaying

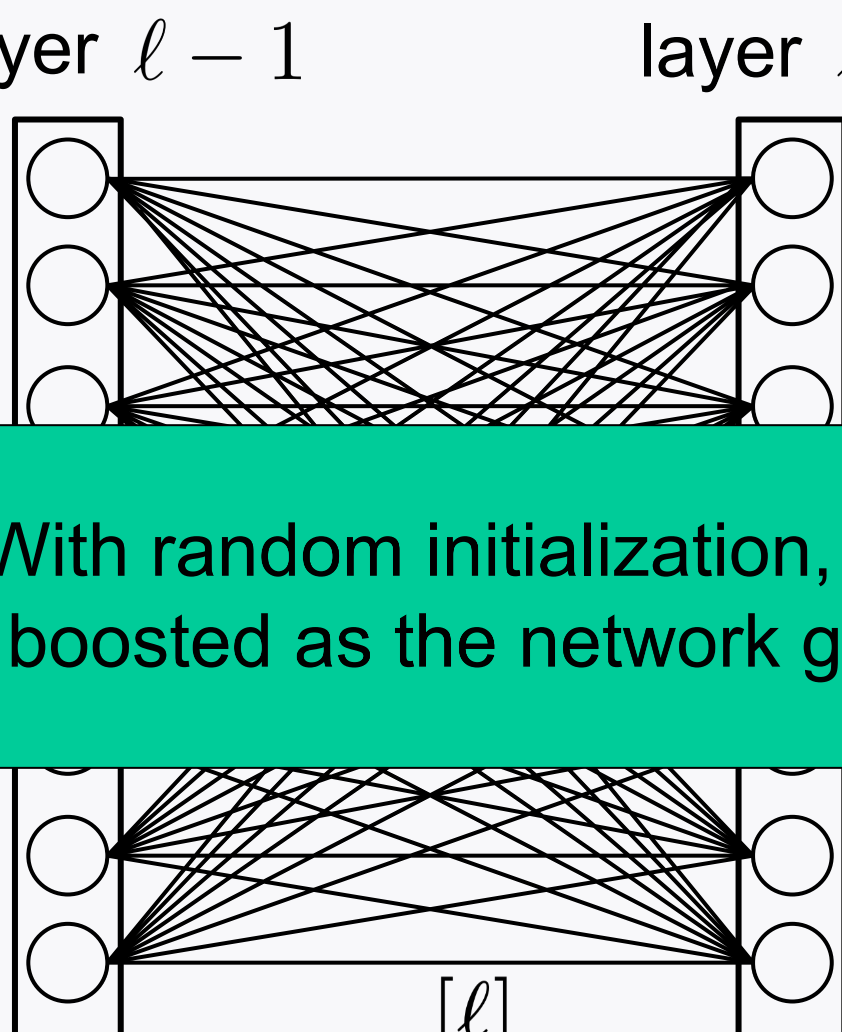
   Batch normalization

Xavier's initialization: Motivation



random initialization?

Xavier's initialization: Motivation



Turns out: With random initialization, a dynamic range of signals is boosted as the network gets deeper.

$w_{ij}^{[\ell]}$ random initialization?

Xavier's initialization: Motivation

To see this "exploding problem", consider:

$$z_1^{[\ell]} = \sum_{j=1}^{n^{[\ell-1]}} w_{1j}^{[\ell]} a_j^{[\ell-1]}$$

A dynamic range of signals can be quantified via:

$$\mathsf{var}\left(z_1^{[\ell]}\right) = \mathsf{var}\left(\sum_{j=1}^{n^{[\ell-1]}} w_{1j}^{[\ell]} a_j^{[\ell-1]}\right)$$

Variance computation

$$\text{var}\left(z_1^{[\ell]}\right) = \text{var}\left(\sum_{j=1}^{n^{[\ell-1]}} w_{1j}^{[\ell]} a_j^{[\ell-1]}\right)$$

$$= \sum_{j=1}^{n^{[\ell-1]}} \text{var}\left(w_{1j}^{[\ell]} a_j^{[\ell-1]}\right)$$

$$= \sum_{j=1}^{n^{[\ell-1]}} \mathbb{E}\left[(w_{1j}^{[\ell]})^2 (a_j^{[\ell-1]})^2\right] - \sum_{j=1}^{n^{[\ell-1]}} \left(\cancelto{0}{\mathbb{E}\left[w_{1j}^{[\ell]} a_j^{[\ell-1]}\right]}\right)^2$$

Assumption:

(i) weights independent

(ii) input independent

(iii) weights/input ind.

(iv) zero mean

Variance computation

$$\mathsf{var}\left(z_1^{[\ell]}\right) = \sum_{j=1}^{n^{[\ell-1]}} \mathbb{E}\left[(w_{1j}^{[\ell]})^2 (a_j^{[\ell-1]})^2\right]$$

$$= \sum_{j=1}^{n^{[\ell-1]}} \mathbb{E}\left[(w_{1j}^{[\ell]})^2\right] \mathbb{E}\left[(a_j^{[\ell-1]})^2\right]$$

$$= \sum_{j=1}^{n^{[\ell-1]}} \mathsf{var}\left(w_{1j}^{[\ell]}\right) \mathsf{var}\left(a_j^{[\ell-1]}\right)$$

Assumption:

(i) weights independent

(ii) input independent

(iii) weights/input ind.

(iv) zero mean

Exploding problem

$$\text{var}\left(z_1^{[\ell]}\right) = \sum_{j=1}^{n^{[\ell-1]}} \text{var}\left(w_{1j}^{[\ell]}\right) \text{var}\left(a_j^{[\ell-1]}\right)$$

Suppose: $\text{var}\left(a_j^{[\ell-1]}\right) = 1, \ \text{var}\left(w_{1j}^{[\ell]}\right) = 1$

Then: $\text{var}\left(z_1^{[\ell]}\right) = n^{[\ell-1]}$

As the network gets deeper, explode!

Xavier's initialization

$$\text{var}\left(z_1^{[\ell]}\right) = \sum_{j=1}^{n^{[\ell-1]}} \text{var}\left(w_{1j}^{[\ell]}\right) \text{var}\left(a_j^{[\ell-1]}\right)$$

Suppose: $\text{var}\left(a_j^{[\ell-1]}\right) = 1$

Idea: Set $\text{var}\left(w_{1j}^{[\ell]}\right) = \frac{1}{n^{[\ell-1]}}$

$$\boxed{w_{ij}^{[\ell]} \text{ i.i.d. } \sim \mathcal{N}\left(0, \frac{1}{n^{[\ell-1]}}\right)}$$

He's initialization: Motivation

$$a_1^{[\ell]} = \mathsf{ReLU}(z_1^{[\ell]})$$
$$= \max(0, z_1^{[\ell]})$$

$$\mathsf{var}\left(a_1^{[\ell]}\right) = \frac{1}{2}\mathsf{var}\left(z_1^{[\ell]}\right)$$

Xavier's initialization $w_{ij}^{[\ell]}$ i.i.d. $\sim \mathcal{N}\left(0, \frac{1}{n^{[\ell-1]}}\right)$

$$\longrightarrow \mathsf{var}\left(a_1^{[\ell]}\right) = \frac{1}{2}\mathsf{var}\left(a_j^{[\ell-1]}\right)$$

He's initialization

$$w_{ij}^{[\ell]} \text{ i.i.d. } \sim \mathcal{N}\left(0, \frac{2}{n^{[\ell-1]}}\right)$$

Techniques for training stability:

Adam optimizer

Learning rate decaying

Batch normalization

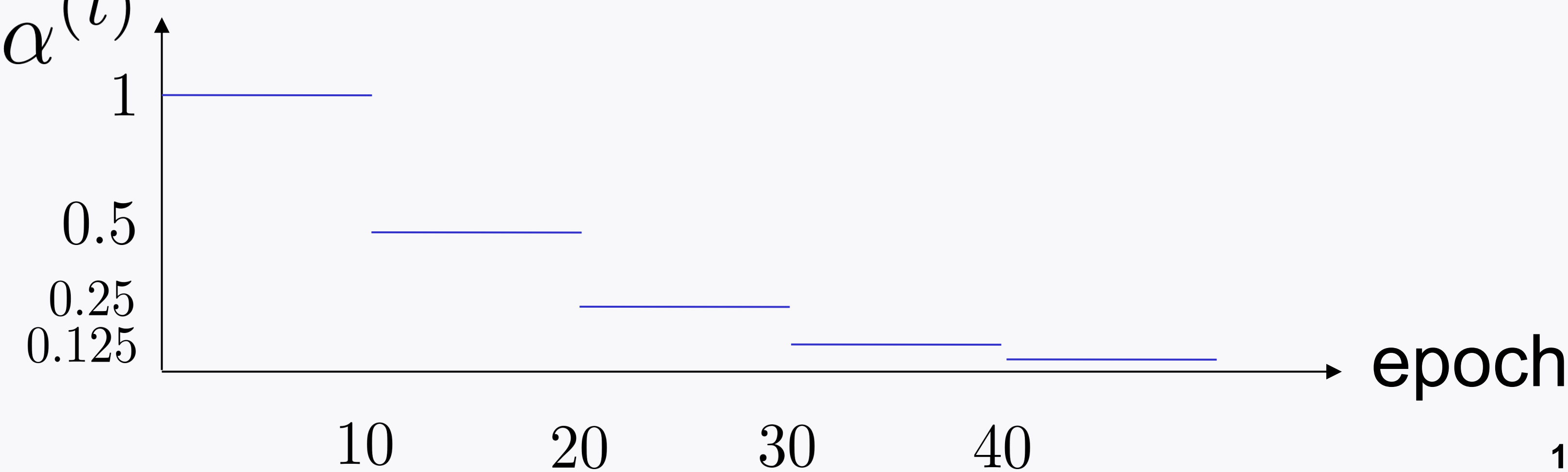
Learning rate decaying

Three popular choices:

1. $\alpha^{(t)} = \gamma^t \qquad 0 < \gamma < 1$

2. $\alpha^{(t)} = \dfrac{1}{\sqrt{t}}$

3. $\alpha^{(t)}$



Batch normalization: Motivation

Turns out: Different signal scalings across distinct layers incur training instability.

One prominent way to address this:

Batch normalization

Batch

Recall the cost function used for gradient descent:

$$J(w^{(t)}) := \frac{1}{m}\sum_{i=1}^{\color{red}{m}} -y^{(i)}\log\hat{y}^{(i)} - (1-y^{(i)})\log(1-\hat{y}^{(i)})$$

Issue: Computationally heavy for a large *m*.

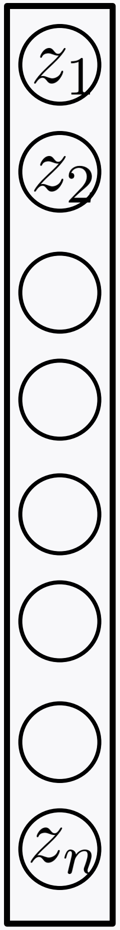
Hence: In practice, use a chunk of examples, called a *batch*.

batch1	batch2	batch3	$\cdots$	

$\underbrace{\text{batch1}}_{m_{\mathcal{B}}}$ examples

Batch normalization

A hidden layer



BN

1. Normalization

component-wise

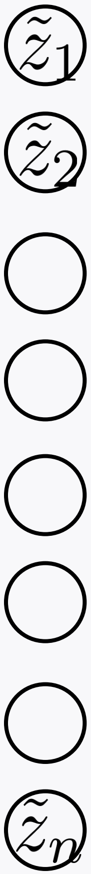
$$z_{\text{norm}} = \frac{z - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

$$\mu_{\mathcal{B}} = \frac{1}{m_{\mathcal{B}}} \sum_{i \in \mathcal{B}} z^{(i)} \quad \sigma_{\mathcal{B}}^2 = \frac{1}{m_{\mathcal{B}}} \sum_{i \in \mathcal{B}} (z^{(i)} - \mu_{\mathcal{B}})^2$$

2. Customized scaling

$$\tilde{z} = \gamma z_{\text{norm}} + \beta$$

$\gamma, \beta \in \mathbf{R}^n$ (learnable parameters)



Look ahead

Will study:

hyperparameter search

cross validation