Deep Learning for Data Science
DS 542

https://dl4ds.github.io/fa2025/

Initialization

Original slides by Tom Gardos,
other content from Understanding Deep Learning unless otherwise cited

Plan for Today

- Project 1
- The need for weights initialization
- Expectations Refresher
- The (Kaiming) He Initialization
- Lottery tickets

Initialization

- Consider standard building block of NN in terms of pre-activations:

new preactivations

$$\mathbf{f}_k = \boldsymbol{\beta}_k + \boldsymbol{\Omega}_k \mathbf{h}_k$$
$$= \boldsymbol{\beta}_k + \boldsymbol{\Omega}_k \mathrm{a}[\mathbf{f}_{k-1}]$$

previous postactivations

- How do we initialize the biases and weights? β_k Ω_k
- Equivalent to choosing starting point in our gradient descent searches

Zero initialization → zeros everywhere, mostly zero gradients

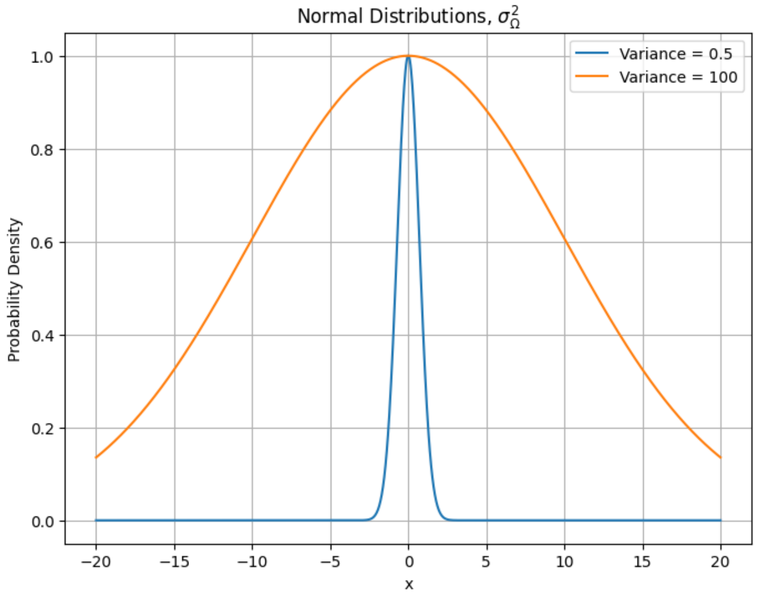
uniform random b/w -1 and 1

→ normal distribution $\mu=0, \sigma^2=1$

Forward Pass

- Consider standard building block of NN in terms of *pre-activations*:

$$\mathbf{f}_k = \boldsymbol{\beta}_k + \boldsymbol{\Omega}_k \mathbf{h}_k$$
$$= \boldsymbol{\beta}_k + \boldsymbol{\Omega}_k \mathrm{a}[\mathbf{f}_{k-1}]$$



- Set all the biases to 0

$$\boldsymbol{\beta}_k = \mathbf{0}$$

- Set weights to be normally distributed
  - mean 0
  - variance σ_Ω^2 will reason about this

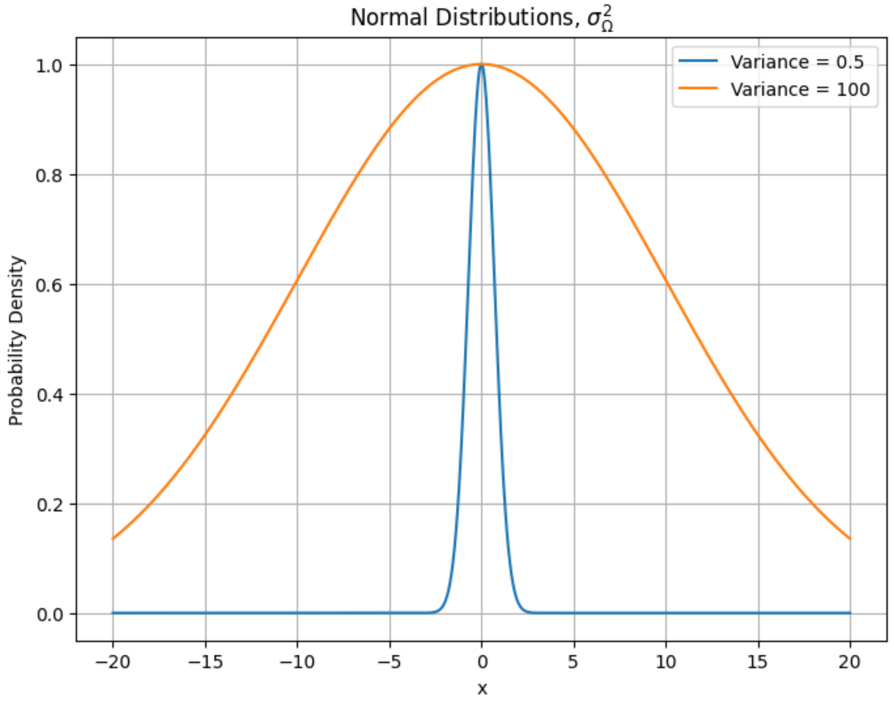
- What will happen as we move through the network if σ_Ω^2 is very small?
- What will happen as we move through the network if σ_Ω^2 is very large?

Backward Pass

$$\frac{\partial \ell_i}{\partial \mathbf{f}_{k-1}} = \mathbb{I}[\mathbf{f}_{k-1} > 0] \odot \left(\mathbf{\Omega}_k^T \frac{\partial \ell_i}{\partial \mathbf{f}_k} \right), \qquad k \in \{K, K-1, \dots 1\} \qquad (7.13)$$

- What will happen as we propagate backwards through the network if σ_Ω^2 is very small?

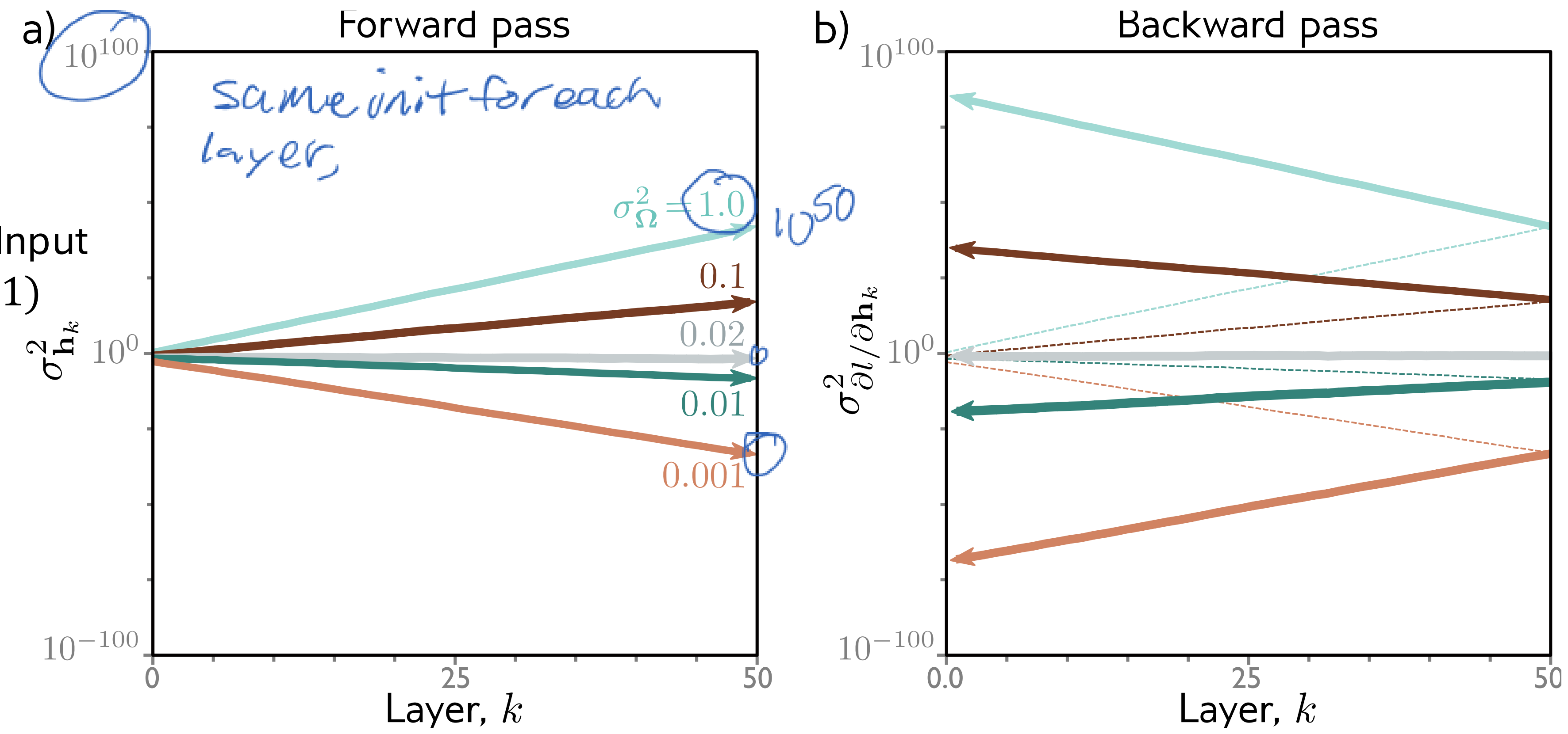
- What will happen as we propagate backwards through the network if σ_Ω^2 is very large?



Initialize weights to different variances

preactivation $= \underbrace{\sum_i h \cdot (\text{norm}) \cdot \sigma}_{\text{is this growing or shrinking?}}$

100D Input
$\sim N(0,1)$



Exploding gradients

Vanishing gradients

average zero input

inputs 100 100 ... 100

Figure 7.4 Weight initialization. Consider a deep network with 50 hidden layers and $D_h = 100$ hidden units per layer. The network has a 100 dimensional input $\mathbf{x}$ initialized with values from a standard normal distribution, a single output fixed at $y = 0$, and a least squares loss function. The bias vectors $\boldsymbol{\beta}_k$ are initialized to zero and the weight matrices $\boldsymbol{\Omega}_k$ are initialized with a normal distribution with mean zero and five different variances $\sigma^2_\Omega \in \{0.001, 0.01, 0.02, 0.1, 1.0\}$. a)

How do we initialize weights to keep variance stable across layers?

Aim: keep variance same between two layers

$$\mathbf{f}' = \boldsymbol{\beta} + \boldsymbol{\Omega}\mathbf{h}$$

$$\mathbf{h} = \mathbf{a}[\mathbf{f}],$$

Definition of variance:

$$\sigma_{f'}^2 = \mathbb{E}[(f_i' - \mathbb{E}[f_i'])^2]$$

Any Questions?

???

- The need for weights initialization
- Expectations Refresher
- The (Kaiming) He initialization
- Lottery tickets

Expectations

continuoes

$$\mathbb{E}\Big[\mathrm{g}[x]\Big] = \int \mathrm{g}[x] Pr(x) dx,$$

Interpretation: what is the average value of g[x] when taking into account the probability of x?

Consider discrete case and assume uniform probability so calculating g[x] reduces to taking average:

$$\mathbb{E}\Big[\mathrm{g}[x]\Big] \approx \frac{1}{N}\sum_{n=1}^{N} \mathrm{g}[x_n^*] \qquad \text{where} \qquad x_n^* \sim Pr(x)$$

discrete

Common Expectation Functions

$\mathbb{E}[(x-\mu)^2] = \text{variance}$

$E[x^k]$

Function g[•]	Expectation
x	mean, μ
x^k	kth moment about zero
$(x-\mu)^k$	kth moment about the mean
$(x-\mu)^2$	variance
$(x-\mu)^3$	skew
$(x-\mu)^4$	kurtosis

$= E[x]$

Table B.1 Special cases of expectation. For some functions $g[x]$, the expectation $\mathbb{E}[g[x]]$ is given a special name. Here we use the notation μ_x to represent the mean with respect to random variable x.

Rules for manipulating expectation

$$\mathbb{E}\Big[k\Big] = k$$ constant

$$\mathbb{E}\Big[k \cdot \mathrm{g}[x]\Big] = k \cdot \mathbb{E}\Big[\mathrm{g}[x]\Big]$$ multiplication by constant

$$\mathbb{E}\Big[\mathrm{f}[x] + \mathrm{g}[x]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big] + \mathbb{E}\Big[\mathrm{g}[x]\Big]$$ addition

$$\mathbb{E}\Big[\mathrm{f}[x]g[y]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big]\mathbb{E}\Big[\mathrm{g}[y]\Big] \qquad \text{if} \quad x, y \quad \text{independent}$$

product

not true if dependent!

Any Questions?

???

- The need for weights initialization
- Expectations Refresher
- The (Kaiming) He initialization
- Lottery tickets

Aim: keep variance same between two layers

$$\mathbf{h} = \mathbf{a}[\mathbf{f}],$$
$$\mathbf{f}' = \boldsymbol{\beta} + \boldsymbol{\Omega}\mathbf{h}$$

Definition of variance:

$$\sigma^2_{f_i'} = \mathbb{E}[(f_i' - \mathbb{E}[f_i'])^2]$$

Now let's prove:

$$\mathbb{E}\left[(x-\mu)^2\right] = \mathbb{E}[x^2] - \mathbb{E}[x]^2$$

useful for proofs,
avoid implementing,
not numerically stable.

variance = expected square − expectation squared

Keeping in mind:

$$\mathbb{E}[x] = \mu$$

Rule 1:	$\mathbb{E}\Big[k\Big] = k$
Rule 2:	$\mathbb{E}\Big[k \cdot \mathrm{g}[x]\Big] = k \cdot \mathbb{E}\Big[\mathrm{g}[x]\Big]$
Rule 3:	$\mathbb{E}\Big[\mathrm{f}[x] + \mathrm{g}[x]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big] + \mathbb{E}\Big[\mathrm{g}[x]\Big]$
Def'n	$\mathbb{E}[x] = \mu$

$$\mathbb{E}[(x-\mu)^2] = \mathbb{E}[x^2 - 2x\mu + \mu^2]$$

just expand

Rule 1: $\mathbb{E}\Big[k\Big] = k$

Rule 2: $\mathbb{E}\Big[k \cdot \mathrm{g}[x]\Big] = k \cdot \mathbb{E}\Big[\mathrm{g}[x]\Big]$

Rule 3: $\mathbb{E}\Big[\mathrm{f}[x] + \mathrm{g}[x]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big] + \mathbb{E}\Big[\mathrm{g}[x]\Big]$

Def'n $\mathbb{E}[x] = \mu$

$$
\begin{aligned}
\mathbb{E}[(x-\mu^2)] &= \mathbb{E}[x^2 - 2x\mu + \mu^2] \\
&= \mathbb{E}[x^2] - \mathbb{E}[2x\mu] + \mathbb{E}[\mu^2]
\end{aligned}
$$

Rule 1: $\mathbb{E}\left[k\right] = k$

Rule 2: $\mathbb{E}\left[k \cdot \mathrm{g}[x]\right] = k \cdot \mathbb{E}\left[\mathrm{g}[x]\right]$

Rule 3: $\mathbb{E}\left[\mathrm{f}[x] + \mathrm{g}[x]\right] = \mathbb{E}\left[\mathrm{f}[x]\right] + \mathbb{E}\left[\mathrm{g}[x]\right]$

Def'n $\mathbb{E}[x] = \mu$

$$
\begin{aligned}
\mathbb{E}[(x - \mu^2)] &= \mathbb{E}[x^2 - 2x\mu + \mu^2] \\
&= \mathbb{E}[x^2] - \mathbb{E}[2x\mu] + \mathbb{E}[\mu^2] \\
&= \mathbb{E}[x^2] - \underbrace{2\mu}_{\text{pulled out constant}}\mathbb{E}[x] + \mu^2
\end{aligned}
$$

Rule 1: $\mathbb{E}\left[k\right] = k$

Rule 2: $\mathbb{E}\left[k \cdot \mathrm{g}[x]\right] = k \cdot \mathbb{E}\left[\mathrm{g}[x]\right]$

Rule 3: $\mathbb{E}\left[\mathrm{f}[x] + \mathrm{g}[x]\right] = \mathbb{E}\left[\mathrm{f}[x]\right] + \mathbb{E}\left[\mathrm{g}[x]\right]$

Def'n $\mathbb{E}[x] = \mu$

$$
\begin{aligned}
\mathbb{E}[(x-\mu^2)] &= \mathbb{E}[x^2 - 2x\mu + \mu^2] \\
&= \mathbb{E}[x^2] - \mathbb{E}[2x\mu] + \mathbb{E}[\mu^2] \\
&= \mathbb{E}[x^2] - 2\mu\underbrace{\mathbb{E}[x]}_{\mu} + \mu^2 \\
&= \mathbb{E}[x^2] - 2\mu^2 + \mu^2
\end{aligned}
$$

Rule 1: $\mathbb{E}\Big[k\Big] = k$

Rule 2: $\mathbb{E}\Big[k \cdot \mathrm{g}[x]\Big] = k \cdot \mathbb{E}\Big[\mathrm{g}[x]\Big]$

Rule 3: $\mathbb{E}\Big[\mathrm{f}[x] + \mathrm{g}[x]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big] + \mathbb{E}\Big[\mathrm{g}[x]\Big]$

Def'n $\mathbb{E}[x] = \mu$

$$
\begin{aligned}
\mathbb{E}[(x-\mu^2)] &= \mathbb{E}[x^2 - 2x\mu + \mu^2] \\
&= \mathbb{E}[x^2] - \mathbb{E}[2x\mu] + \mathbb{E}[\mu^2] \\
&= \mathbb{E}[x^2] - 2\mu\mathbb{E}[x] + \mu^2 \\
&= \mathbb{E}[x^2] - 2\mu^2 + \mu^2 \\
&= \mathbb{E}[x^2] - \mu^2
\end{aligned}
$$

Rule 1: $\mathbb{E}\Big[k\Big] = k$

Rule 2: $\mathbb{E}\Big[k \cdot \mathrm{g}[x]\Big] = k \cdot \mathbb{E}\Big[\mathrm{g}[x]\Big]$

Rule 3: $\mathbb{E}\Big[\mathrm{f}[x] + \mathrm{g}[x]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big] + \mathbb{E}\Big[\mathrm{g}[x]\Big]$

Def'n $\mathbb{E}[x] = \mu$

$$
\begin{aligned}
\mathbb{E}[(x-\mu^2)] &= \mathbb{E}[x^2 - 2x\mu + \mu^2] \\
&= \mathbb{E}[x^2] - \mathbb{E}[2x\mu] + \mathbb{E}[\mu^2] \\
&= \mathbb{E}[x^2] - 2\mu\mathbb{E}[x] + \mu^2 \\
&= \mathbb{E}[x^2] - 2\mu^2 + \mu^2 \\
&= \mathbb{E}[x^2] - \mu^2 \\
&= \mathbb{E}[x^2] - E[x]^2
\end{aligned}
$$

Aim: keep variance same between two layers

$$\mathbf{f}' = \boldsymbol{\beta} + \boldsymbol{\Omega}\mathbf{h}$$

$$\mathbf{h} = \mathbf{a}[\mathbf{f}],$$

$$\sigma^2_{f'} = \mathbb{E}[(f'_i - \mathbb{E}[f'_i])^2]$$

direct variance formula

$$\sigma^2_{f'} = \mathbb{E}\left[f'^2_i\right] - \mathbb{E}[f'_i]^2$$

alternate variance just proven

$$\longrightarrow \quad \mathbb{E}\left[(x-\mu)^2\right] = \mathbb{E}[x^2] - \mathbb{E}[x]^2$$

Aim: keep variance same between two layers

$$\mathbf{f}' = \boldsymbol{\beta} + \boldsymbol{\Omega}\mathbf{h}$$

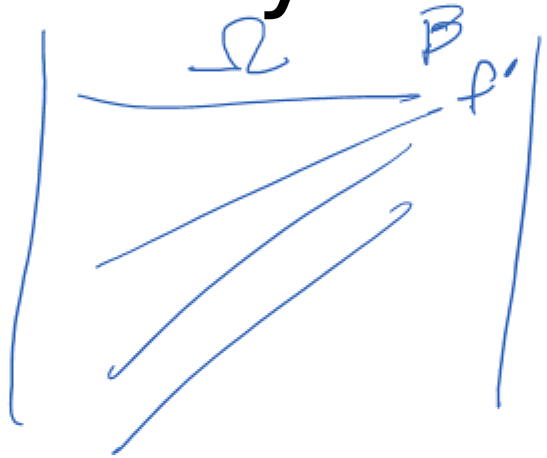
$$\mathbf{h} = \mathbf{a}[\mathbf{f}],$$

$$\sigma_{f'}^2 = \mathbb{E}[(f_i' - \mathbb{E}[f_i'])^2]$$

$$\sigma_{f'}^2 = \mathbb{E}\left[f_i'^2\right] - \mathbb{E}[f_i']^2$$

Focus on this term.

Aim: keep variance same between two layers



$$\mathbf{f}' = \boldsymbol{\beta} + \boldsymbol{\Omega}\mathbf{h}$$

Consider the mean of the pre-activations:

$$\mathbb{E}[f_i'] = \mathbb{E}\left[\beta_i + \sum_{j=1}^{D_h} \Omega_{ij} h_j\right]$$

looking at one
specific unit/value.

Rule 1: $\mathbb{E}\left[k\right] = k$

Rule 2: $\mathbb{E}\left[k \cdot \mathrm{g}[x]\right] = k \cdot \mathbb{E}\left[\mathrm{g}[x]\right]$

Rule 3: $\mathbb{E}\left[\mathrm{f}[x] + \mathrm{g}[x]\right] = \mathbb{E}\left[\mathrm{f}[x]\right] + \mathbb{E}\left[\mathrm{g}[x]\right]$

Rule 4: $\mathbb{E}\left[\mathrm{f}[x]g[y]\right] = \mathbb{E}\left[\mathrm{f}[x]\right]\mathbb{E}\left[\mathrm{g}[y]\right] \quad \text{if} \quad x, y \quad \text{independent}$

$$\mathbb{E}[f_i'] = \mathbb{E}\left[\beta_i + \sum_{j=1}^{D_h} \Omega_{ij} h_j\right]$$

$$= \mathbb{E}\left[\beta_i\right] + \sum_{j=1}^{D_h} \mathbb{E}\left[\Omega_{ij} h_j\right]$$

Rule 1: $\mathbb{E}\Big[k\Big] = k$

Rule 2: $\mathbb{E}\Big[k \cdot \mathrm{g}[x]\Big] = k \cdot \mathbb{E}\Big[\mathrm{g}[x]\Big]$

Rule 3: $\mathbb{E}\Big[\mathrm{f}[x] + \mathrm{g}[x]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big] + \mathbb{E}\Big[\mathrm{g}[x]\Big]$

Rule 4: $\mathbb{E}\Big[\mathrm{f}[x]g[y]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big]\mathbb{E}\Big[\mathrm{g}[y]\Big] \quad \text{if} \quad x, y \quad \text{independent}$

$$
\begin{aligned}
\mathbb{E}[f_i'] &= \mathbb{E}\left[\beta_i + \sum_{j=1}^{D_h} \Omega_{ij} h_j\right] \\
&= \mathbb{E}\left[\beta_i\right] + \sum_{j=1}^{D_h} \mathbb{E}\left[\Omega_{ij} h_j\right] \\
&= \mathbb{E}\left[\beta_i\right] + \sum_{j=1}^{D_h} \mathbb{E}\left[\Omega_{ij}\right] \mathbb{E}\left[h_j\right]
\end{aligned}
$$

independence.

Rule 1: $\mathbb{E}\Big[k\Big] = k$

Rule 2: $\mathbb{E}\Big[k \cdot \mathrm{g}[x]\Big] = k \cdot \mathbb{E}\Big[\mathrm{g}[x]\Big]$

Rule 3: $\mathbb{E}\Big[\mathrm{f}[x] + \mathrm{g}[x]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big] + \mathbb{E}\Big[\mathrm{g}[x]\Big]$

Rule 4: $\mathbb{E}\Big[\mathrm{f}[x]g[y]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big]\mathbb{E}\Big[\mathrm{g}[y]\Big] \quad \text{if} \quad x, y \quad \text{independent}$

$$
\begin{aligned}
\mathbb{E}[f_i'] &= \mathbb{E}\left[\beta_i + \sum_{j=1}^{D_h} \Omega_{ij} h_j\right] \\
&= \mathbb{E}[\beta_i] + \sum_{j=1}^{D_h} \mathbb{E}[\Omega_{ij} h_j] \\
&= \mathbb{E}[\beta_i] + \sum_{j=1}^{D_h} \mathbb{E}[\Omega_{ij}]\, \mathbb{E}[h_j] \\
&= 0 + \sum_{j=1}^{D_h} 0 \cdot \mathbb{E}[h_j] = 0
\end{aligned}
$$

Start making initialization choices.

- Set all the biases to 0
- Weights normally distributed
  - mean 0
  - variance σ_Ω^2

Aim: keep variance same between two layers

$$\mathbf{f}' = \boldsymbol{\beta} + \boldsymbol{\Omega}\mathbf{h}$$

$$\mathbf{h} = \mathbf{a}[\mathbf{f}],$$

$$\sigma_{f'}^2 = \mathbb{E}[(f_i' - \mathbb{E}[f_i'])^2]$$

$$\sigma_{f'}^2 = \mathbb{E}\left[f_i'^2\right] - \underbrace{\mathbb{E}[f_i']}_{0}{}^2 = \mathbb{E}\left[f_i'^2\right]$$

Rule 1: $\mathbb{E}\Big[k\Big] = k$

Rule 2: $\mathbb{E}\Big[k \cdot \mathrm{g}[x]\Big] = k \cdot \mathbb{E}\Big[\mathrm{g}[x]\Big]$

Rule 3: $\mathbb{E}\Big[\mathrm{f}[x] + \mathrm{g}[x]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big] + \mathbb{E}\Big[\mathrm{g}[x]\Big]$

Rule 4: $\mathbb{E}\Big[\mathrm{f}[x]g[y]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big]\mathbb{E}\Big[\mathrm{g}[y]\Big] \quad \text{if} \quad x, y \quad \text{independent}$

$$
\begin{aligned}
\sigma^2_{f'} &= \mathbb{E}[f_i'^2] - \mathbb{E}[f_i']^2 \\
&= \mathbb{E}\left[\left(\beta_i + \sum_{j=1}^{D_h} \Omega_{ij} h_j\right)^2\right] - 0
\end{aligned}
$$

Set all the biases to 0

Weights normally distributed

mean 0

variance σ^2_Ω

Rule 1: $\mathbb{E}\Big[k\Big] = k$

Rule 2: $\mathbb{E}\Big[k \cdot \mathrm{g}[x]\Big] = k \cdot \mathbb{E}\Big[\mathrm{g}[x]\Big]$

Rule 3: $\mathbb{E}\Big[\mathrm{f}[x] + \mathrm{g}[x]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big] + \mathbb{E}\Big[\mathrm{g}[x]\Big]$

Rule 4: $\mathbb{E}\Big[\mathrm{f}[x]g[y]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big]\mathbb{E}\Big[\mathrm{g}[y]\Big] \quad \text{if} \quad x, y \quad \text{independent}$

$$
\begin{aligned}
\sigma^2_{f'} &= \mathbb{E}[f_i'^2] - \mathbb{E}[f_i']^2 \\
&= \mathbb{E}\left[\left(\beta_i + \sum_{j=1}^{D_h} \Omega_{ij} h_j\right)^2\right] - 0 \\
&= \mathbb{E}\left[\left(\sum_{j=1}^{D_h} \Omega_{ij} h_j\right)^2\right]
\end{aligned}
$$

Initialization choices.

- Set all the biases to 0
- Weights normally distributed
  - mean 0
  - variance σ^2_Ω

Rule 1: $\mathbb{E}\Big[k\Big] = k$

Rule 2: $\mathbb{E}\Big[k \cdot \mathrm{g}[x]\Big] = k \cdot \mathbb{E}\Big[\mathrm{g}[x]\Big]$

Rule 3: $\mathbb{E}\Big[\mathrm{f}[x] + \mathrm{g}[x]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big] + \mathbb{E}\Big[\mathrm{g}[x]\Big]$

Rule 4: $\mathbb{E}\Big[\mathrm{f}[x]g[y]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big]\mathbb{E}\Big[\mathrm{g}[y]\Big] \quad \text{if} \quad x, y \quad \text{independent}$

On diagonal of $(\Sigma)^2$, $(\Omega_{i,j})^2 \cdot h_j^2$

Off diagonal $\Omega_{i,j_1} h_{j_1} \Omega_{i,j_2} h_{j_2}$

$\Omega_{ij} h_j$

multiplyall pairs.

$$\sigma^2_{f'} = \mathbb{E}[f_i'^2] - \mathbb{E}[f_i']^2$$

$$= \mathbb{E}\left[\left(\beta_i + \sum_{j=1}^{D_h} \Omega_{ij} h_j\right)^2\right] - 0$$

$$= \mathbb{E}\left[\left(\sum_{j=1}^{D_h} \Omega_{ij} h_j\right)^2\right]$$

$$= \sum_{j=1}^{D_h} \mathbb{E}\left[\Omega_{ij}^2\right] \mathbb{E}\left[h_j^2\right]$$

$E[(\Omega_{ij} h_j)^2]$ but separated b/c independent

For all the cross terms, $E[\Omega_{ij}] = 0$ so only the squared terms are left, then use independence.

Initialization choices.

independent, each mean $\varnothing$, so off diagonal entries have mean zero

- Set all the biases to 0
- Weights normally distributed
  - mean 0
  - variance σ^2_Ω

Rule 1: $\mathbb{E}\Big[k\Big] = k$

Rule 2: $\mathbb{E}\Big[k\cdot \mathrm{g}[x]\Big] = k\cdot\mathbb{E}\Big[\mathrm{g}[x]\Big]$

Rule 3: $\mathbb{E}\Big[\mathrm{f}[x]+\mathrm{g}[x]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big]+\mathbb{E}\Big[\mathrm{g}[x]\Big]$

Rule 4: $\mathbb{E}\Big[\mathrm{f}[x]g[y]\Big] = \mathbb{E}\Big[\mathrm{f}[x]\Big]\mathbb{E}\Big[\mathrm{g}[y]\Big] \quad \text{if} \quad x,y \quad \text{independent}$

$$
\begin{aligned}
\sigma^2_{f'} &= \mathbb{E}[f_i'^2] - \mathbb{E}[f_i']^2 \\
&= \mathbb{E}\left[\left(\beta_i + \sum_{j=1}^{D_h}\Omega_{ij}h_j\right)^2\right] - 0 \\
&= \mathbb{E}\left[\left(\sum_{j=1}^{D_h}\Omega_{ij}h_j\right)^2\right] \\
&= \sum_{j=1}^{D_h}\mathbb{E}\left[\Omega_{ij}^2\right]\mathbb{E}\left[h_j^2\right] \\
&= \sum_{j=1}^{D_h}\sigma^2_{\Omega}\mathbb{E}\left[h_j^2\right] = \sigma^2_{\Omega}\sum_{j=1}^{D_h}\mathbb{E}\left[h_j^2\right]
\end{aligned}
$$

last slide

Because the Ω's are zero mean, this is the variance.

Initialization choices.

- Set all the biases to 0
- Weights normally distributed
  - mean 0
  - variance σ^2_{Ω}

$$\sigma^2_{f'} = \sigma^2_\Omega \sum_{j=1}^{D_h} \mathbb{E}\left[h_j^2\right]$$

last slide.

hj = postactivation, insert its formula

$$= \sigma^2_\Omega \sum_{j=1}^{D_h} \mathbb{E}\left[\mathrm{ReLU}[f_j]^2\right]$$

assumed activation function.

$$= \sigma^2_\Omega \sum_{j=1}^{D_h} \int_{-\infty}^{\infty} \mathrm{ReLU}[f_j]^2 Pr(f_j) df_j$$

From the definition of expectation.

ReLU definition

$$= \sigma^2_\Omega \sum_{j=1}^{D_h} \int_{-\infty}^{\infty} (\underbrace{\mathbb{I}[f_j > 0]}_{1 \text{ if } f_j \geq 0,\ 0 \text{ otherwise.}} f_j)^2 Pr(f_j) df_j$$

$$= \sigma^2_\Omega \sum_{j=1}^{D_h} \int_{0}^{\infty} f_j^2 Pr(f_j) df_j$$

Only positive integral limits because of ReLU

half of continuous variance calc

f_j has zero mean

$$= \sigma^2_\Omega \sum_{j=1}^{D_h} \frac{\sigma_f^2}{2} = \frac{D_h \sigma^2_\Omega \sigma^2_f}{2}$$

half variance

½ of the variance for zero mean distribution

Aim: keep variance same between two layers

Since:

$$\sigma^2_{f'} = \frac{D_h \sigma^2_\Omega \sigma^2_f}{2}$$

So $\frac{D_h \sigma^2_\Omega}{2} = 1$

Should choose:

$$\sigma^2_\Omega = \frac{2}{D_h}$$

use this variance for init.

To get:

$$\sigma^2_{f'} = \sigma^2_f$$

Kaiming He 何恺明

https://people.csail.mit.edu/kaiming/

This is called He initialization or Kaiming initialization.

K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," Proc. IEEE International Conference on Computer Vision, 2015, pp. 1026–1034. Accessed: Feb. 11, 2024.

He initialization (assumes ReLU)

- Forward pass: want the variance of hidden unit activations in layer k+1 to be the same as variance of activations in layer k:

$$\sigma_{\Omega}^2 = \frac{2}{D_h}$$

Number of units at layer k

- Backward pass: want the variance of gradients at layer k to be the same as variance of gradient in layer k+1:

Should all layers have same width?

$$\sigma_{\Omega}^2 = \frac{2}{D_{h'}}$$

Number of units at layer k+1

different layers!

$\sigma_{\Omega}^2 = \frac{4}{D_h + D_{h'}}$ if $D_h \neq D_{h'}$ (heuristic, not theory)

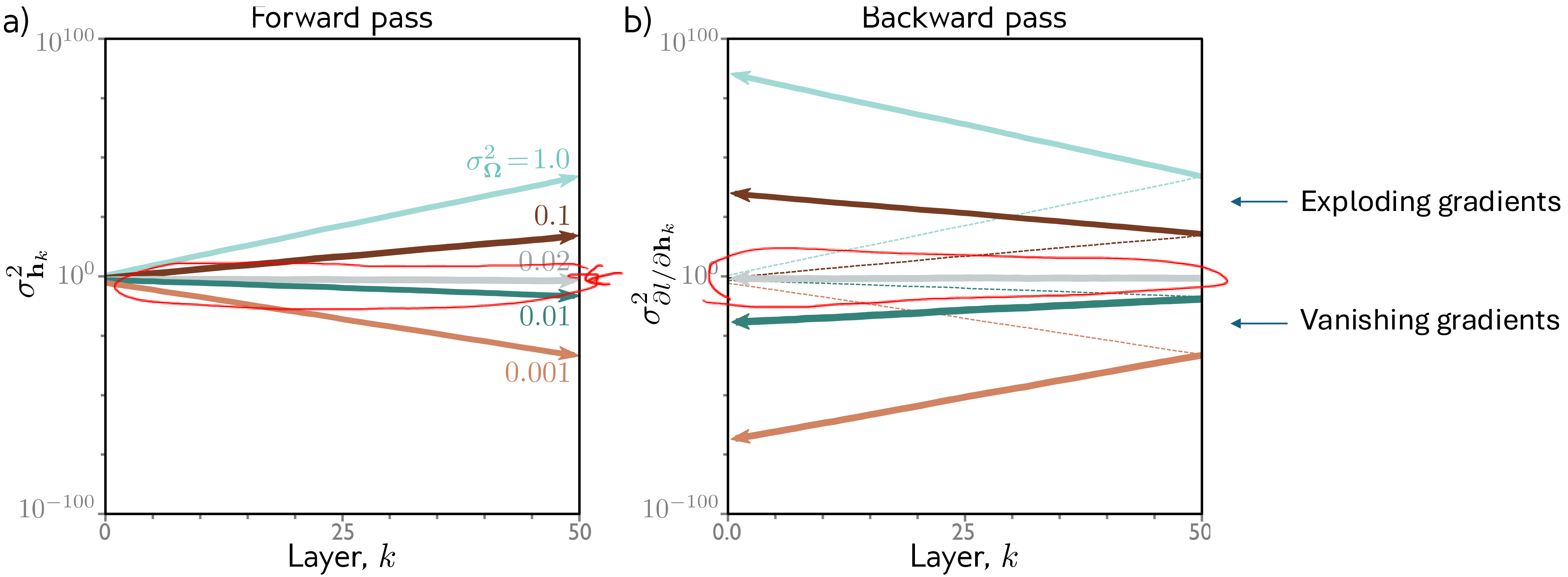


Figure 7.4 Weight initialization. Consider a deep network with 50 hidden layers and $D_h = 100$ hidden units per layer. The network has a 100 dimensional input $\mathbf{x}$ initialized with values from a standard normal distribution, a single output fixed at $y = 0$, and a least squares loss function. The bias vectors $\boldsymbol{\beta}_k$ are initialized to zero and the weight matrices $\boldsymbol{\Omega}_k$ are initialized with a normal distribution with mean zero and five different variances $\sigma^2_{\boldsymbol{\Omega}} \in \{0.001, 0.01, 0.02, 0.1, 1.0\}$. a)

$$\boxed{\sigma^2_{\Omega} = \frac{2}{D_h} = \frac{2}{100} = 0.02}$$

Default Initialization in PyTorch

https://pytorch.org/docs/stable/nn.init.html#torch.nn.init.kaiming_uniform_

Does not match what we just analyzed.

torch.nn.init.kaiming_uniform_(*tensor*, *a=0*, *mode='fan_in'*, *nonlinearity='leaky_relu'*, *generator=None*) [SOURCE]

Fill the input *Tensor* with values using a Kaiming uniform distribution.

The method is described in *Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification* - He, K. et al. (2015). The resulting tensor will have values sampled from $\mathcal{U}(-\text{bound}, \text{bound})$ where

$$\text{bound} = \text{gain} \times \sqrt{\frac{3}{\text{fan\_mode}}}$$

Also known as He initialization.

? does not match

Any Questions?

???

- The need for weights initialization
- Expectations Refresher
- The (Kaiming) He initialization
- Lottery tickets

Initialization Note

A good initialization does not prevent gradient descent from changing the weights a lot.

- A good initialization keeps the initial gradients modestly sized,
- And modest gradients reduce wild swings in parameters with gradient descent
- Smaller learning rates also help with this.
- Next week's topic, regularization, will directly address this.

next Monday

Limitations of Initialization

- No guarantees that the model will train to low losses
- No guarantees that training process won't lead to large values or gradients
- No guarantees that the model won't have lots of inactive units
    - In fact, the estimates adjusted for half being inactive!

- In fact, much of the network is often useless, and could be pruned away!

The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks

Neural network pruning techniques can reduce the parameter counts of trained networks by over 90%, decreasing storage requirements and improving computational performance of inference without compromising accuracy. However, contemporary experience is that the sparse architectures produced by pruning are difficult to train from the start, which would similarly improve training performance.

We find that a standard pruning technique naturally uncovers subnetworks whose initializations made them capable of training effectively. Based on these results, we articulate the "lottery ticket hypothesis:" dense, randomly-initialized, feed-forward networks contain subnetworks ("winning tickets") that - when trained in isolation - reach test accuracy comparable to the original network in a similar number of iterations. The winning tickets we find have won the initialization lottery: their connections have initial weights that make training particularly effective.

We present an algorithm to identify winning tickets and a series of experiments that support the lottery ticket hypothesis and the importance of these fortuitous initializations. We consistently find winning tickets that are less than 10-20% of the size of several fully-connected and convolutional feed-forward architectures for MNIST and CIFAR10. Above this size, the winning tickets that we find learn faster than the original network and reach higher test accuracy.

Any Questions?

???

- The need for weights initialization
- Expectations Refresher
- The (Kaiming) He initialization
- Lottery tickets

Disclaimer

- Just because variance of gradients starts the same does not mean that the variance of gradients stays the same.

- You should still check the gradients if you are having training difficulties…

Bonus Tip

- If you are trying to implement a model based on a paper, and you are having trouble training, check if they shared their code.
  - Many papers omit important initialization details. 🕵️
  - Especially if they say that their method is not sensitive to initialization. 🤔
  - Also, some paper descriptions of initialization don't match their code. 🤬