CPEN455: Deep Learning
Homework 1

Seiya Nozawa-Temchenko

Finished: Jan. 31, 2025

1 Dropout [25pts]

Let us consider a single hidden layer MLP with M hidden units. Suppose the input vector $\mathbf{x} \in \mathbb{R}^{N\times 1}$. The hidden activations $\mathbf{h} \in \mathbb{R}^{M\times 1}$ are computed as follows,

$$\mathbf{h} = \sigma(W\mathbf{x} + \mathbf{b}) \tag{1}$$

where weight matrix $W \in \mathbb{R}^{N\times M}$, bias vector $\mathbf{b} \in \mathbb{R}^{M\times 1}$, and σ is the nonlinear activation function.

Dropout [1] is a technique to help reduce overfitting for neural networks. In PyTorch, Dropout is implemented as follows. During training, we independently zero out elements of $\mathbf{h}$ with probability p and then rescale it with $\frac{1}{1-p}$, *i.e.*,

$$\tilde{\mathbf{h}} = \frac{\mathbf{m}}{1-p} \odot \mathbf{h} \tag{2}$$

$$\mathbf{m}[i] \sim \text{Bernoulli}(1-p) \quad \forall i = 1, ..., M, \tag{3}$$

where $\odot$ is the Hadamard product (*a.k.a*, element-wise product and Bernoulli$(1-p)$ is the Bernoulli distribution where the random variable takes the value 1 with the probability $1-p$. During testing, we just use $\tilde{\mathbf{h}} = \mathbf{h}$.

1.1 [5pts] Explain why we need to rescale the hidden activations by $\frac{1}{1-p}$.

Rescaling the hidden activation from $\mathbf{h}$ to $\tilde{\mathbf{h}}$ by $\frac{1}{1-p}$ during training is done to compensate for the sum reduction caused by Dropout. Dropout randomly zeroes out a fraction p of the hidden activation, reducing the total contribution of the hidden layer to the next layer. Without rescaling, the network's output would be smaller during training than during testing.

1.2 [15pts] Assume $\mathbf{x} \sim \mathcal{N}(0, I)$, $\mathbf{b} = 0$, $WW^\top = I_M$ (I_M is an identity matrix with size $M \times M$), and we use rectified linear units (ReLU) as the nonlinear activation function, *i.e.*, $\sigma(x) = \max(x, 0)$, derive the variance of the activations before Dropout (*i.e.*, $\mathbf{h}$) and after Dropout (*i.e.*, $\tilde{\mathbf{h}}$).

Since $\mathbf{b} = 0$, $WW^\top = I_M$, and $\mathbf{x} \sim \mathcal{N}(0, I)$, each element of $W\mathbf{x}$ is distributed as $\mathcal{N}(0, 1)$. Let pre-activation values $z_i \to z \sim \mathcal{N}(0, 1)$ for the sake of simplicity,

$$\mathbf{h} = \text{ReLU}(W\mathbf{x}) = \max(W\mathbf{x}, 0) \to h_i = \text{ReLU}[z] \tag{4}$$

To compute Var[ReLU(z)], we must calculate the expectations:

$$\text{Var}[\text{ReLU}(z)] = \mathbb{E}[\text{ReLU}^2(z)] - \mathbb{E}^2[\text{ReLU}(z)] \tag{5}$$

$$\mathbb{E}[\text{ReLU}(z)] = \int_{-\infty}^{\infty} \text{ReLU}(z) f_z(z)\, dz \tag{6}$$

We can use the PDF for a standard normal distribution $f_z(z)$ defined as:

$$f_z(z) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{z^2}{2}) \tag{7}$$

Since ReLU zeroes negative values, we can evaluate the expectation:

$$\begin{aligned} \mathbb{E}[\text{ReLU}(z)] &= \int_0^{\infty} z \frac{1}{\sqrt{2\pi}} \exp(-\frac{z^2}{2})\, dz \\ u = -\frac{z^2}{2} &\rightarrow du = -z dz \\ \mathbb{E}[\text{ReLU}(z)] &= -\frac{1}{\sqrt{2\pi}} \int_0^{\infty} \exp(u)\, du \\ &= -\frac{1}{\sqrt{2\pi}} \exp(-\frac{z^2}{2})\Big|_0^{\infty} = \frac{1}{\sqrt{2\pi}} \end{aligned} \tag{8}$$

Now we can compute square expectation which can be calculated using the concept of symmetry:

$$\mathbb{E}[\text{ReLU}^2(z)] = \int_0^{\infty} z^2 \frac{1}{\sqrt{2\pi}} \exp(-\frac{z^2}{2})\, dz \tag{9}$$

$$\mathbb{E}[z^2] = \text{Var}(z) + \mathbb{E}^2[z] = 1 + 0 = 1 \tag{10}$$

$$\begin{aligned} \mathbb{E}[z^2 \mathbb{1}(z > 0)] = \mathbb{E}[z^2 \mathbb{1}(z < 0)] &= \frac{1}{2}\mathbb{E}[z^2] = \frac{1}{2} \\ \therefore \mathbb{E}[\text{ReLU}^2(z)] &= \frac{1}{2} \end{aligned} \tag{11}$$

We can now find the variance before Dropout by evaluating Var[ReLU(z)] from (5):

$$\begin{aligned} \text{Var}[\text{ReLU}(z)] &= \mathbb{E}[\text{ReLU}^2(z)] - \mathbb{E}^2[\text{ReLU}(z)] \\ \text{Var}[h_i] &= \frac{1}{2} - \frac{1}{\sqrt{2\pi}} = \frac{\pi - 1}{2\pi} \end{aligned} \tag{12}$$

Now we can find the variance after Dropout.

$$\tilde{h_i} = \frac{m_i}{1-p} h_i \quad m_i \sim \text{Bernoulli}(1-p) \tag{13}$$

$$\text{Var}[aX] = a^2 \text{Var}[X],\ \text{Var}[XY] = \text{Var}[X]\text{Var}[Y] + \text{Var}[X]\mathbb{E}^2[Y] + \text{Var}[Y]\mathbb{E}^2[X] \tag{14}$$

$$\mathbb{E}[m_i] = 1 - p,\ \text{Var}[m_i] = p(1-p),\ \mathbb{E}[h_i] = \frac{1}{\sqrt{2\pi}},\ \text{Var}[h_i] = \frac{\pi - 1}{2\pi} \tag{15}$$

$$\begin{aligned}
\text{Var}[\tilde{h_i}] &= (\frac{1}{1-p})^2 \text{Var}[m_i h_i] \\
&= (\frac{1}{1-p})^2 [p(1-p)(\frac{\pi-1}{2\pi}) + p(1-p)(\frac{1}{\sqrt{2\pi}})^2 + (\frac{\pi-1}{2\pi})(1-p)^2] \\
&= (\frac{1}{1-p})^2 [(1-p)(\frac{\pi-1+p\pi}{2\pi})] \\
\text{Var}[\tilde{h_i}] &= \frac{\pi-1+p(2-\pi)}{2\pi(1-p)}
\end{aligned} \tag{16}$$

1.3 [5pts] What is the expected number of hidden units that are kept (*i.e.*, those with $\mathbf{m}[i] = 1$) by Dropout? Derive the probability distribution (*i.e.*, the probability mass function) of the number of kept hidden units.

Each hidden unit is kept with $1-p$ or dropped with p. With M hidden units, the expected units kept can be found by:

$$\mathbb{E}[K] = \sum_{i=1}^{M} \mathbb{E}[\mathbf{m}[i]] = \sum_{i=1}^{M} (1-p) = M(1-p) \tag{17}$$

Since each hidden unit is either kept or dropped and each unit is kept independently of others, the number of hidden units K follows a binomial distribution. The PMF for K is then given by choosing k units of M, the probability of keeping k units, and the probability of dropping the remaining $M-k$ units:

$$K \sim \text{Binomial}(M, 1-p) \tag{18}$$

$$\mathbb{P}(K=k) = \binom{M}{k}(1-p)^k p^{M-k}, \quad \binom{M}{k} = \frac{M!}{k!(M-k)!} \tag{19}$$

1.4 [Bonus 10pts] Assume the number of hidden units M goes to infinity and the probability of keeping units $1-p$ goes to 0 in a way that their product $M(1-p)$ stays fixed. Derive the probability distribution of the number of kept hidden units.

The number of kept units K follows a Binomial distribution in Eq. (18), with PMF in Eq. (19) for some $k = 0, 1, 2, ..., M$. We can apply the Poisson limit theorem that the Binomial distribution converges to a Poisson distribution since $M \to \infty$ and $1-p \to 0$ with $M(1-p) = \lambda$ held constant.

$$\lambda = M(1-p) \to p = 1 - \frac{\lambda}{M} \quad \therefore p^M = (1 - \frac{\lambda}{M})^M \tag{20}$$

$$\lim_{M\to\infty} \left(1 - \frac{\lambda}{M}\right)^M = e^{-\lambda} \quad \therefore p^M \approx e^{-\lambda} \tag{21}$$

Since k is fixed and M is large, we can then approximate p^{M-k} and $\binom{M}{k}$:

$$p^{M-k} \approx p^M \approx e^{-\lambda} = e^{-M(1-p)} \tag{22}$$

$$\binom{M}{k} = \frac{M!}{k!(M-k)!} \approx \frac{M^k}{k!} \tag{23}$$

$$\begin{aligned}\mathbb{P}(K=k) &\approx \frac{M^k}{k!}(1-p)^k e^{-M(1-p)} \\ &= \frac{(M(1-p))^k e^{-M(1-p)}}{k!} = \frac{\lambda^k e^{-\lambda}}{k!}\end{aligned} \tag{24}$$

The number of hidden units kept K then follows a Poisson distribution:

$$K \sim \text{Poisson}(\lambda),\ \lambda = M(1-p) \tag{25}$$

1.5 [Bonus 10pts] Suppose the number of hidden units M follows a Poisson distribution with parameter λ, *i.e.*, the probability mass function is $\mathbb{P}(M=k) = \frac{\lambda^k e^{-\lambda}}{k!}$. Derive the probability distribution of the number of kept hidden units.

The number of hidden units M follows a Poisson distribution with λ:

$$\mathbb{P}(M=m) = \frac{\lambda^m e^{-\lambda}}{m!},\ m = 0, 1, 2, ... \tag{26}$$

Given $M=m$, the number of kept units $K|M=m$ follows a Binomial distribution:

$$K|M=m \sim \text{Binomial}(m, 1-p) \tag{27}$$

$$\mathbb{P}(K=k|M=m) = \binom{m}{k}(1-p)^k p^{m-k},\ k = 0, 1, 2, ..., m \tag{28}$$

We can now derive the marginal (unconditional) distribution of K:

$$\begin{aligned}\mathbb{P}(K=k) &= \sum_{m=k}^{\infty} \mathbb{P}(K=k|M=m)\mathbb{P}(M=m) \\ &= \sum_{m=k}^{\infty} \binom{m}{k}(1-p)^k p^{m-k} \frac{\lambda^m e^{-\lambda}}{m!},\ \binom{m}{k} = \frac{m!}{k!(m-k)!} \\ &= \frac{(1-p)^k e^{-\lambda}}{k!} \sum_{m=k}^{\infty} \frac{(\lambda p)^{m-k}}{(m-k)!}\end{aligned} \tag{29}$$

Let $n = m - k$ to use the following Taylor series expansion for $e^{\lambda p}$ to simplify our expression:

$$\sum_{m=k}^{\infty} \frac{(\lambda p)^{m-k}}{(m-k)!} = \sum_{n=0}^{\infty} \frac{(\lambda p)^n}{n!} = e^{\lambda p} \tag{30}$$

$$\mathbb{P}(K=k) = \frac{(1-p)^k e^{-\lambda}}{k!} e^{\lambda p} = \frac{(1-p)^k e^{-\lambda(1-p)}}{k!} \tag{31}$$

We can incorporate the following simplification to align with the Poisson PMF:

$$(1-p)^k = \left(\frac{\lambda(1-p)}{\lambda}\right)^k = \left(\frac{\lambda'}{\lambda}\right)^k \to (\lambda')^k = (\lambda(1-p))^k \tag{32}$$

$$\therefore \mathbb{P}(K=k) = \frac{(\lambda(1-p))^k e^{-\lambda(1-p)}}{k!} = \frac{(\lambda')^k e^{-\lambda'}}{k!} \tag{33}$$

This is the PMF of a Poisson distribution $K \sim \text{Poisson}(\lambda')$ with parameter $\lambda' = \lambda(1-p)$.

2 Batch Normalization [35pts]

Let us again consider a single hidden layer MLP with M hidden units. Suppose we now have B input vectors packed as a matrix such that each row is a sample, *i.e.*, $X \in \mathbb{R}^{B\times N}$. The hidden activations $H \in \mathbb{R}^{B\times M}$ are computed as follows,

$$Y = XW^\top + \mathbf{b}^\top \tag{34}$$

$$H = \sigma(Y) \tag{35}$$

where weight matrix $W \in \mathbb{R}^{M\times N}$, bias vector $\mathbf{b} \in \mathbb{R}^{M\times 1}$, and σ is the nonlinear activation function. $XW^\top + \mathbf{b}^\top$ leverages the *broadcasting addition*. In particular, the shapes of $XW^\top$ and $\mathbf{b}^\top$ are $B \times M$ and $1 \times M$, respectively. You can imagine that the broadcasting addition first duplicates $b^\top$ for B times to create a matrix of shape $B \times M$ and then performs element-wise addition. Note that in the actual implementation, one does not perform duplication since it is a waste of memory. This is just a helpful way to understand how broadcasting operations work in a high level.

Batch normalization (BN) [2], is a method that makes training of deep neural networks faster and more stable via normalizing the inputs of individual layers by re-centering and re-scaling. In particular, if we apply BN to pre-activations Y in Eq. (34), we have

$$\mathbf{m} = \frac{1}{B}\sum_{i=1}^{B} Y[i,:] \tag{36}$$

$$\mathbf{v}[j] = \frac{1}{B}\sum_{i=1}^{B}(Y[i,:] - \mathbf{m}[j])^2 \tag{37}$$

$$\hat{Y}[i,j] = \boldsymbol{\gamma}[j]\frac{Y[i,j] - \mathbf{m}[j]}{\sqrt{\mathbf{v}[j] + \epsilon}} + \boldsymbol{\beta}[j] \tag{38}$$

where the normalized input $\hat{Y} \in \mathbb{R}^{B\times M}$. $\mathbf{m} \in \mathbb{R}^{M\times 1}$ and $\mathbf{v} \in \mathbb{R}^{M\times 1}$ are the mean and the (dimension-wise) variance vectors. $\boldsymbol{\gamma} \in \mathbb{R}^{M\times 1}$ and $\boldsymbol{\beta} \in \mathbb{R}^{M\times 1}$ are learnable parameters associated with the BN layer. ϵ is a hyperparameter.

2.1 [5pts] Explain why we need the hyperparameter ϵ

The purpose of hyperparater ϵ in BN is for numerical stability or, more specifically, to prevent division by 0. Therefore, it is typically a very small number. It also plays a small role in the overall standardization process, enabling the BN layer to normalize the inputs.

2.2 [10pts] Derive the mean and variance of $\hat{Y}[i,j]$ (ignore the effects of ϵ only for this question)

We can begin with deriving the mean of $\hat{Y}[i,j]$. Given Eq. (38), by definition of $\mathbf{m}[j]$, we can find expectation $\mathbb{E}[Y[i,j]]$:

$$\mathbf{m}[j] = \frac{1}{B}\sum_{i=1}^{B} Y[i,j] \;\; \therefore \mathbb{E}[Y[i,j]] = \mathbf{m}[j] \tag{39}$$

Knowing this equivalence and assuming $\epsilon = 0$, we can find the mean of $\hat{Y}[i,j]$. Note that the mean transformation property allows that $\mathbb{E}[aX + b] = a\mathbb{E}[X] + b$.

$$\mathbb{E}[Y[i,j] - \mathbf{m}[j]] = \mathbb{E}[Y[i,j]] - \mathbf{m}[j] = 0 \tag{40}$$

$$\mathbb{E}\left[\frac{Y[i,j] - \mathbf{m}[j]}{\sqrt{\mathbf{v}[j]}}\right] = \frac{1}{\sqrt{\mathbf{v}[j]}}\mathbb{E}[Y[i,j] - \mathbf{m}[j]] = 0 \tag{41}$$

$$\mathbb{E}[\hat{Y}[i,j]] = \boldsymbol{\gamma}[j]\mathbb{E}\left[\frac{Y[i,j] - \mathbf{m}[j]}{\sqrt{\mathbf{v}[j]}}\right] + \boldsymbol{\beta}[j] = 0 + \boldsymbol{\beta}[j] = \boldsymbol{\beta}[j] \tag{42}$$

We can find the variance of $\hat{Y}[i,j]$. Note that while $\boldsymbol{\gamma}[j]$ and $\boldsymbol{\beta}[j]$ are learnable parameters, which can be treated as constants during forward pass, they are not random variables in this context. Also note that the variance transformation property allows that $\mathrm{Var}[aX + b] = a^2\mathrm{Var}[X]$.

$$\mathbf{v}[j] = \frac{1}{B}\sum_{i=1}^{B}(Y[i,j] - \mathbf{m}[j])^2 \;\; \therefore \mathrm{Var}[Y[i,j]] = \mathbf{v}[j] \tag{43}$$

$$\mathrm{Var}[Y[i,j] - \mathbf{m}[j]] = \mathrm{Var}[Y[i,j]] = \mathbf{v}[j] \tag{44}$$

$$\mathrm{Var}\left[\frac{Y[i,j] - \mathbf{m}[j]}{\sqrt{\mathbf{v}[j]}}\right] = \frac{1}{\mathbf{v}[j]}\mathrm{Var}[Y[i,j] - \mathbf{m}[j]] = \frac{\mathbf{v}[j]}{\mathbf{v}[j]} = 1 \tag{45}$$

$$\mathrm{Var}[\hat{Y}[i,j]] = \mathrm{Var}\left[\boldsymbol{\gamma}[j]\frac{Y[i,j] - \mathbf{m}[j]}{\sqrt{\mathbf{v}[j]}} + \boldsymbol{\beta}[j]\right] = \boldsymbol{\gamma}^2[j]\mathrm{Var}\left[\frac{Y[i,j] - \mathbf{m}[j]}{\sqrt{\mathbf{v}[j]}}\right] = \boldsymbol{\gamma}^2[j] \tag{46}$$

2.3 [20pts] Suppose the nonlinear activation function is ReLU and the gradient of the training loss ℓ with respect to (w.r.t.) the normalized hidden activations H is $\frac{\partial \ell}{\partial H}$. Derive the gradients of the training loss w.r.t. learnable parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, the mean $\mathbf{m}$, the variance $\mathbf{v}$, and the input Y.

To derive the gradients of the training loss ℓ with respect to $\boldsymbol{\gamma}$, $\boldsymbol{\beta}$, $\mathbf{m}$, $\mathbf{v}$, and Y, we can use the chain rule. We are given BN transformation in Eq. (38) and the activation function is:

$$H[i,j] = \mathrm{ReLU}(\hat{Y}[i,j]) = \max(0, \hat{Y}[i,j]) \tag{47}$$

First, we will start with finding the gradients with respect to the learnable scale parameter $\boldsymbol{\gamma}$:

$$\frac{\partial \ell}{\partial \boldsymbol{\gamma}[j]} = \sum_{i=1}^{B}\frac{\partial \ell}{\partial H[i,j]}\frac{\partial H[i,j]}{\partial \boldsymbol{\gamma}[j]} \tag{48}$$

To compute $\frac{\partial H[i,j]}{\partial \boldsymbol{\gamma}[j]}$, we can refer to $H[i,j] = \mathrm{ReLU}(\hat{Y}[i,j])$:

$$\frac{\partial H[i,j]}{\partial \boldsymbol{\gamma}[j]} = \frac{\partial H[i,j]}{\partial \hat{Y}[i,j]}\frac{\partial \hat{Y}[i,j]}{\partial \boldsymbol{\gamma}[j]} \tag{49}$$

$$\frac{\partial H[i,j]}{\partial \hat{Y}[i,j]} = \mathbb{1}(\hat{Y}[i,j] > 0) \tag{50}$$

Note that the indicator function $\mathbb{1}(\hat{Y}[i,j] > 0)$ is a Boolean function to represent gradient flow activation for when $\hat{Y}[i,j]$ is positive.

$$\frac{\partial \hat{Y}[i,j]}{\partial \boldsymbol{\gamma}[j]} = \frac{Y[i,j] - \mathbf{m}[j]}{\sqrt{\mathbf{v}[j] + \epsilon}} \tag{51}$$

$$\frac{\partial H[i,j]}{\partial \boldsymbol{\gamma}[j]} = \mathbb{1}(\hat{Y}[i,j] > 0) \frac{Y[i,j] - \mathbf{m}[j]}{\sqrt{\mathbf{v}[j] + \epsilon}} \tag{52}$$

$$\frac{\partial \ell}{\partial \boldsymbol{\gamma}[j]} = \sum_{i=1}^{B} \frac{\partial \ell}{\partial H[i,j]} \mathbb{1}(\hat{Y}[i,j] > 0) \frac{Y[i,j] - \mathbf{m}[j]}{\sqrt{\mathbf{v}[j] + \epsilon}} \tag{53}$$

Second, we will find the gradient with respect to learnable shift parameter $\boldsymbol{\beta}$:

$$\frac{\partial \ell}{\partial \boldsymbol{\beta}[j]} = \sum_{i=1}^{B} \frac{\partial \ell}{\partial H[i,j]} \frac{\partial H[i,j]}{\partial \boldsymbol{\beta}[j]} \tag{54}$$

To compute $\frac{\partial H[i,j]}{\partial \boldsymbol{\beta}[j]}$, we can use a similar equation to Eq. (49) and refer to Eq. (50) for the derivative of the ReLU function:

$$\frac{\partial H[i,j]}{\partial \boldsymbol{\beta}[j]} = \frac{\partial H[i,j]}{\partial \hat{Y}[i,j]} \frac{\partial \hat{Y}[i,j]}{\partial \boldsymbol{\beta}[j]} \tag{55}$$

$$\frac{\partial \hat{Y}[i,j]}{\partial \boldsymbol{\beta}[j]} = 1 \;\; \therefore \frac{\partial H[i,j]}{\partial \boldsymbol{\beta}[j]} = \mathbb{1}(\hat{Y}[i,j] > 0) \tag{56}$$

$$\frac{\partial \ell}{\partial \boldsymbol{\beta}[j]} = \sum_{i=1}^{B} \frac{\partial \ell}{\partial H[i,j]} \mathbb{1}(\hat{Y}[i,j] > 0) \tag{57}$$

Now we can find the gradient with respect to mean $\mathbf{m}[j]$ by starting with the chain rule:

$$\frac{\partial \ell}{\partial \mathbf{m}[j]} = \sum_{k=1}^{B} \frac{\partial \ell}{\partial \hat{Y}[k,j]} \frac{\partial \hat{Y}[k,j]}{\partial \mathbf{m}[j]} \tag{58}$$

$$\frac{\partial \ell}{\partial \hat{Y}[k,j]} = \frac{\partial \ell}{\partial H[k,j]} \mathbb{1}(\hat{Y}[k,j] > 0) \tag{59}$$

We can now compute $\frac{\partial \hat{Y}[k,j]}{\partial \mathbf{m}[j]}$ by differentiating the BN transformation in Eq. (38):

$$\frac{\partial \hat{Y}[k,j]}{\partial \mathbf{m}[j]} = \boldsymbol{\gamma}[j] \frac{\partial}{\partial \mathbf{m}[j]} \left(\frac{Y[k,j] - \mathbf{m}[j]}{\sqrt{\mathbf{v}[j] + \epsilon}} \right) = -\frac{\boldsymbol{\gamma}[j]}{\sqrt{\mathbf{v}[j] + \epsilon}} \tag{60}$$

$$\begin{aligned} \frac{\partial \ell}{\partial \mathbf{m}[j]} &= \sum_{k=1}^{B} \frac{\partial \ell}{\partial \hat{Y}[k,j]} \left(-\frac{\boldsymbol{\gamma}[j]}{\sqrt{\mathbf{v}[j] + \epsilon}} \right) \\ &= -\frac{\boldsymbol{\gamma}[j]}{\sqrt{\mathbf{v}[j] + \epsilon}} \sum_{k=1}^{B} \frac{\partial \ell}{\partial H[k,j]} \mathbb{1}(\hat{Y}[k,j] > 0) \end{aligned} \tag{61}$$

Next, we can find the gradient with respect to variance $\mathbf{v}[j]$ by starting with the chain rule:

$$\frac{\partial \ell}{\partial \mathbf{v}[j]} = \sum_{k=1}^{B} \frac{\partial \ell}{\partial \hat{Y}[k,j]} \frac{\partial \hat{Y}[k,j]}{\partial \mathbf{v}[j]} \tag{62}$$

$$\frac{\partial \ell}{\partial \hat{Y}[k,j]} = \frac{\partial \ell}{\partial H[k,j]} \mathbb{1}(\hat{Y}[k,j] > 0) \tag{63}$$

We can now compute $\frac{\partial \hat{Y}[k,j]}{\partial \mathbf{v}[j]}$ by differentiating the BN transformation in Eq. (38):

$$\frac{\partial \hat{Y}[k,j]}{\partial \mathbf{v}[j]} = \boldsymbol{\gamma}[j] \frac{\partial}{\partial \mathbf{v}[j]} \left(\frac{Y[k,j] - \mathbf{m}[j]}{\sqrt{\mathbf{v}[j] + \epsilon}} \right) = \boldsymbol{\gamma}[j](Y[k,j] - \mathbf{m}[j]) \left(-\frac{1}{2} (\mathbf{v}[j] + \epsilon)^{-\frac{3}{2}} \right) \tag{64}$$

$$\begin{aligned} \frac{\partial \ell}{\partial \mathbf{v}[j]} &= \sum_{k=1}^{B} \frac{\partial \ell}{\partial \hat{Y}[k,j]} \left(-\frac{\boldsymbol{\gamma}[j](Y[k,j] - \mathbf{m}[j])}{2(\mathbf{v}[j] + \epsilon)^{\frac{3}{2}}} \right) \\ &= -\frac{\boldsymbol{\gamma}[j]}{2(\mathbf{v}[j] + \epsilon)^{\frac{3}{2}}} \sum_{k=1}^{B} (Y[k,j] - \mathbf{m}[j]) \frac{\partial \ell}{\partial H[k,j]} \mathbb{1}(\hat{Y}[k,j] > 0) \end{aligned} \tag{65}$$

Lastly, we can find the gradient with respect to pre-activation inputs Y:

$$\frac{\partial \ell}{\partial Y[i,j]} = \sum_{k=1}^{B} \frac{\partial \ell}{\partial \hat{Y}[k,j]} \frac{\partial \hat{Y}[k,j]}{\partial Y[i,j]} \tag{66}$$

$$\frac{\partial \ell}{\partial \hat{Y}[k,j]} = \frac{\partial \ell}{\partial H[k,j]} \mathbb{1}(\hat{Y}[k,j] > 0) \tag{67}$$

$$\frac{\partial \ell}{\partial Y[k,j]} = \sum_{k=1}^{B} \frac{\partial \ell}{\partial H[k,j]} \mathbb{1}(\hat{Y}[k,j] > 0) \frac{\partial \hat{Y}[k,j]}{\partial Y[k,j}} \tag{68}$$

Each pre-activation output $\hat{Y}[k,j]$ depends on all $Y[\ell,j]$ in the batch through both the mean $\mathbf{m}[j]$ and the variance $\mathbf{v}[j]$. So when we compute $\frac{\partial \hat{Y}[k,j]}{\partial Y[i,j]}$, we must account for both direct and indirect dependencies. A direct dependency refers to when $k = i$, that $Y[k,j]$ directly affects $\hat{Y}[k,j]$. Indirect dependency refers to all $Y[k,j]$ affecting $\hat{Y}[k,j]$ through $\mathbf{m}[j]$ and $\mathbf{v}[j]$.

$$\frac{\partial \hat{Y}[k,j]}{\partial Y[i,j]} = \boldsymbol{\gamma}[j] \frac{\partial}{\partial Y[i,j]} \left(\frac{Y[k,j] - \mathbf{m}[j]}{\sqrt{\mathbf{v}[j] + \epsilon}} \right) \tag{69}$$

$$\frac{\partial}{\partial Y[i,j]} \left(\frac{Y[k,j] - \mathbf{m}[j]}{\sqrt{\mathbf{v}[j] + \epsilon}} \right) = \frac{\frac{\partial}{\partial Y[i,j]} (Y[k,j] - \mathbf{m}[j])}{\sqrt{\mathbf{v}[j] + \epsilon}} + (Y[k,j] - \mathbf{m}[j]) \frac{\partial}{\partial Y[i,j]} \left(\frac{1}{\sqrt{\mathbf{v}[j] + \epsilon}} \right) \tag{70}$$

$$\frac{\partial}{\partial Y[i,j]} (Y[k,j] - \mathbf{m}[j]) = \delta(k = i) - \frac{1}{B}, \ \delta(k = i) = \begin{cases} 1, & k = i \\ 0, & k \neq i \end{cases} \tag{71}$$

$$\begin{aligned}\frac{\partial}{\partial Y[i,j]}\left(\frac{1}{\sqrt{\mathbf{v}[j]+\epsilon}}\right) &= -\frac{1}{2}(\mathbf{v}[j]+\epsilon)^{-\frac{3}{2}}\frac{\partial \mathbf{v}[j]}{\partial Y[i,j]} \\ &= -\frac{1}{2}(\mathbf{v}[j]+\epsilon)^{-\frac{3}{2}}\frac{2}{B}(Y[i,j]-\mathbf{m}[j]) \\ &= -\frac{Y[i,j]-\mathbf{m}[j]}{B(\mathbf{v}[j]+\epsilon)^{\frac{3}{2}}}\end{aligned} \tag{72}$$

$$\frac{\partial \hat{Y}[k,j]}{\partial Y[i,j]} = \boldsymbol{\gamma}[j]\left(\frac{\delta(k=i)-\frac{1}{B}}{\sqrt{\mathbf{v}[j]+\epsilon}} - \frac{(Y[k,j]-\mathbf{m}[j])(Y[i,j]-\mathbf{m}[j])}{B(\mathbf{v}[j]+\epsilon)^{\frac{3}{2}}}\right) \tag{73}$$

$$\begin{aligned}\frac{\partial \ell}{\partial Y[i,j]} &= \sum_{k=1}^{B}\frac{\partial \ell}{\partial H[k,j]}\mathbb{1}(\hat{Y}[k,j]>0)\boldsymbol{\gamma}[j]\left(\frac{\delta(k=i)-\frac{1}{B}}{\sqrt{\mathbf{v}[j]+\epsilon}} - \frac{(Y[k,j]-\mathbf{m}[j])(Y[i,j]-\mathbf{m}[j])}{B(\mathbf{v}[j]+\epsilon)^{\frac{3}{2}}}\right) \\ &= \boldsymbol{\gamma}[j]\left(\sum_{k=1}^{B}\frac{\partial \ell}{\partial H[k,j]}\mathbb{1}(\hat{Y}[k,j]>0)\frac{\delta(k=i)-\frac{1}{B}}{\sqrt{\mathbf{v}[j]+\epsilon}}\right. \\ &\quad \left. -\sum_{k=1}^{B}\frac{\partial \ell}{\partial H[k,j]}\mathbb{1}(\hat{Y}[k,j]>0)\frac{(Y[k,j]-\mathbf{m}[j])(Y[i,j]-\mathbf{m}[j])}{B(\mathbf{v}[j]+\epsilon)^{\frac{3}{2}}}\right)\end{aligned} \tag{74}$$

We can now focus on simplifying the expression:

$$\begin{aligned}&\sum_{k=1}^{B}\frac{\partial \ell}{\partial H[k,j]}\mathbb{1}(\hat{Y}[k,j]>0)\left(\delta(k=i)-\frac{1}{B}\right) \\ &= \sum_{k=1}^{B}\frac{\partial \ell}{\partial H[k,j]}\mathbb{1}(\hat{Y}[k,j]>0)\delta(k=i) - \sum_{k=1}^{B}\frac{\partial \ell}{\partial H[k,j]}\mathbb{1}(\hat{Y}[k,j]>0)\frac{1}{B}\end{aligned} \tag{75}$$

$$\sum_{k=1}^{B}\frac{\partial \ell}{\partial H[k,j]}\mathbb{1}(\hat{Y}[k,j]>0)\delta(k=i) = \frac{\partial \ell}{\partial H[i,j]}\mathbb{1}(\hat{Y}[k=i,j]>0) \tag{76}$$

$$\begin{aligned}&\sum_{k=1}^{B}\frac{\partial \ell}{\partial H[k,j]}\mathbb{1}(\hat{Y}[k,j]>0)\frac{(Y[k,j]-\mathbf{m}[j])(Y[i,j]-\mathbf{m}[j])}{B(\mathbf{v}[j]+\epsilon)^{\frac{3}{2}}} \\ &= \frac{Y[i,j]-\mathbf{m}[j]}{B(\mathbf{v}[j]+\epsilon)^{\frac{3}{2}}}\sum_{k=1}^{B}\frac{\partial \ell}{\partial H[k,j]}\mathbb{1}(\hat{Y}[k,j]>0)(Y[k,j]-\mathbf{m}[j])\end{aligned} \tag{77}$$

$$\begin{aligned}\frac{\partial \ell}{\partial Y[i,j]} =& \frac{\boldsymbol{\gamma}[j]}{\sqrt{\mathbf{v}[j]+\epsilon}}\frac{\partial \ell}{\partial H[i,j]}\mathbb{1}(\hat{Y}[i,j]>0) - \frac{\boldsymbol{\gamma}[j]}{B\sqrt{\mathbf{v}[j]+\epsilon}}\sum_{k=1}^{B}\frac{\partial \ell}{\partial H[k,j]}\mathbb{1}(\hat{Y}[k,j]>0) \\ &- \frac{\boldsymbol{\gamma}[j](Y[i,j]-\mathbf{m}[j])}{B(\mathbf{v}[j]+\epsilon)^{\frac{3}{2}}}\sum_{k=1}^{B}\frac{\partial \ell}{\partial H[k,j]}\mathbb{1}(\hat{Y}[k,j]>0)(Y[k,j]-\mathbf{m}[j])\end{aligned} \tag{78}$$

2.4 [Bonus 10pts] BN only normalizes the hidden units in a dimension-wise fashion, *e.g.*, the variance is computed per-dimension. Derive the equations of BN where the covariance matrix of the normalized input $\hat{Y}$ is normalized to an identity. Compare this version of BN to the original

BN and discuss the pros and cons.

Standard BN normalizes each dimension independently, ensuring that each feature has zero mean and unit variance. However, because the normalization is done per-dimension, the off-diagonal elements between different dimensions are not removed. Basically, the covariance matrix of the BN-transformed activations is diagonal with ones on the diagonal but it is not necessarily the identity matrix since the off-diagonal entries or covariances may be non-zero.

In order to normalize the full covariance matrix to the identity, we must de-correlate the dimensions and re-scale them. This can be done with a full whitening transformation by using Eq. (36), Eq. (37), and Eq. (38):

$$\boldsymbol{\mu} = \frac{1}{B}\sum_{i=1}^{B} Y[i,:] \ \in \mathbb{R}^M \tag{79}$$

$$\boldsymbol{\Sigma} = \frac{1}{B}\sum_{i=1}^{B} (Y[i,:] - \boldsymbol{\mu})(Y[i,:] - \boldsymbol{\mu})^\top \ \in \mathbb{R}^{M\times M} \tag{80}$$

We can compute the whitening matrix $\boldsymbol{\Sigma}^{-\frac{1}{2}}$ and then apply the whitening transformation to the centered activations:

$$\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-\frac{1}{2}} = I \tag{81}$$

$$\tilde{Y}[i,:] = \boldsymbol{\Sigma}^{-\frac{1}{2}}(Y[i,:] - \boldsymbol{\mu}) \tag{82}$$

To allow the network to learn, we introduce learnable parameters similar to standard BN (where Γ is a diagonal matrix or vector of scale parameters):

$$\hat{Y}[i,:] = \boldsymbol{\Gamma}\tilde{Y}[i,:] + \boldsymbol{\beta} = \boldsymbol{\Gamma}\boldsymbol{\Sigma}^{-\frac{1}{2}}(Y[i,:] - \boldsymbol{\mu}) + \boldsymbol{\beta} \tag{83}$$

Standard BN:

- Normalizes each feature independently with zero mean and unit variance
- Does not remove inter-feature correlations, so the covariance matrix remains diagonal but not necessarily the identity
- Computationally efficient and easy to implement

Whitening BN:

- Normalizes the full covariance matrix to the identity by removing correlations
- Reduces redundancy and provides better-conditioned inputs for subsequent layers
- More computationally expensive due to $\Sigma^{-\frac{1}{2}}$ and can cause numerical instability when Σ is ill-conditioned

3 Back-Propagation and Initialization [40pts]

Suppose we have a deep MLP classifier and L hidden layers as follows.

$$\mathbf{h}_i = \sigma(W_i \mathbf{h}_{i-1} + \mathbf{b}_i) \quad \forall i = 1, ..., L \tag{84}$$

where the input data vector $\mathbf{h}_0 = \mathbf{x} \in \mathbb{R}^{D_0 \times 1}$ and σ is the nonlinear activation function. Weights and the bias vector of the i-th layer are $W_i \in \mathbb{R}^{D_i \times D_{i-1}}$ and $\mathbf{b}_i \in \mathbb{R}^{D_i \times 1}$ respectively. The readout function is the *softmax*. In other words, the probability of the input $\mathbf{x}$ belong to the class k is,

$$\mathbb{P}(\mathbf{x} \text{ belongs to class } k) = \mathbf{y}[k] = \frac{\exp(\mathbf{h}_L[k])}{\sum_j \exp(\mathbf{h}_L[k])} \tag{85}$$

where the input to the softmax $\mathbf{h}_L[k]) \in \mathbb{R}^{D_L \times 1}$ is often called *logits*. The dimension of the logits should be the same as the total number of classes K, *i.e.*, $D_L = K$. To train the classifier, we use cross-entropy loss as below,

$$\ell(\bar{\mathbf{y}}, \mathbf{y}) = -\sum_{k=1}^{K} \bar{\mathbf{y}}[k] \log \mathbf{y}[k] \tag{86}$$

where $\bar{\mathbf{y}} \in \mathbb{R}^{K \times 1}$ is the ground-truth probability vector of the input $\mathbf{x}$. Typically, it is a one-hot encoded vector, *i.e.*, if input $\mathbf{x}$ of the training set belongs to the class c, then $\bar{\mathbf{y}}[c] = 1$ and $\bar{\mathbf{y}}[k] = 0, \forall k \neq c$.
N.B.: For simplicity, we only consider the case of a single sample in the above context, *i.e.*, the batch dimension is ignored.

3.1 [15pts] Derive the gradients of loss ℓ w.r.t. all hidden activations $\mathbf{h}_i$. In particular, for any layer $1 \leq i \leq L$, you need to show how to derive $\frac{\partial \ell}{\partial \mathbf{h}_i}$. You need to write down the shapes of all tensors involved in the final expressions.

To begin with the gradient at the output (layer L), we must first note that the softmax and cross-entropy loss have the following property. Note that $\mathbf{y} \in \mathbb{R}^{K \times 1}$ is the vector of softmax outputs and $\bar{\mathbf{y}} \in \mathbb{R}^{K \times 1}$ is the one-hot encoded ground-truth label.

$$\frac{\partial \ell}{\partial \mathbf{h}_L} = \mathbf{y} - \bar{\mathbf{y}} \quad \frac{\partial \ell}{\partial \mathbf{h}_L} \in \mathbb{R}^{K \times 1} \tag{87}$$

To find the backpropagation to the hidden layers, for any hidden layer i:

$$\mathbf{h}_i = \sigma(W_i \mathbf{h}_{i-1} + \mathbf{b}_i) \tag{88}$$

To obtain $\frac{\partial \ell}{\partial \mathbf{h}_i}$ in terms of the given variables, we can compute layer $i+1$ as:

$$\mathbf{h}_{i+1} = \sigma(W_{i+1} \mathbf{h}_i + \mathbf{b}_{i+1}) \tag{89}$$

Then by the chain rule, the derivative of the loss with respect to $\mathbf{h}_i$ is obtained by backpropagating through the linear transformation and the nonlinearity of layer $i+1$.

$$\frac{\partial \ell}{\partial \mathbf{h}_i} = W_{i+1}^\top \left[\frac{\partial \ell}{\partial \mathbf{h}_{i+1}} \odot \sigma'(W_{i+1} \mathbf{h}_i + \mathbf{b}_{i+1}) \right] \tag{90}$$

Since $W_{i+1} \in \mathbb{R}^{D_{i+1} \times D_i}$, that $W_{i+1}^\top \in \mathbb{R}^{D_i \times D_{i+1}}$. Also since $\frac{\partial \ell}{\partial \mathbf{h}_{i+1}} \in \mathbb{R}^{D_{i+1} \times 1}$, $\sigma'(W_{i+1}\mathbf{h}_i + \mathbf{b}_{i+1}) \in \mathbb{R}^{D_{i+1} \times 1}$, their element-wise product $\odot$ is in $\mathbb{R}^{D_{i+1} \times 1}$, and therefore the final product makes $\frac{\partial \ell}{\partial \mathbf{h}_i} \in \mathbb{R}^{D_i \times 1}$.

3.2 [10pts] Derive the gradients of loss ℓ w.r.t. all weights W_i and $\mathbf{b}_i$. In particular, for any layer $1 \leq i \leq L$, you need to show how to derive $\frac{\partial \ell}{\partial W_i}$ and $\frac{\partial \ell}{\partial \mathbf{b}_i}$. You need to write down the shapes of all tensors involved in the final expressions.

Since W_i appears in the argument of σ through term $W_i\mathbf{h}_{i-1} + \mathbf{b}_i$, we use the chain rule. We can notice that the derivative of the term $W_i\mathbf{h}_{i-1}$ with respect to W_i is the input $\mathbf{h}_{i-1}$ such that:

$$\frac{\partial \ell}{\partial W_i} = \left[\frac{\partial \ell}{\partial \mathbf{h}_i} \odot \sigma'(W_i\mathbf{h}_{i-1} + \mathbf{b}_i)\right] \mathbf{h}_{i-1}^\top \tag{91}$$

Since $\frac{\partial \ell}{\partial \mathbf{h}_i} \in \mathbb{R}^{D_i \times 1}$, $\sigma'(W_i\mathbf{h}_{i-1} + \mathbf{b}_i \in \mathbb{R}^{D_i \times 1}$, their element-wise product is in $\mathbb{R}^{D_i \times 1}$. With $\mathbf{h}_{i-1}^\top \in \mathbb{R}^{1 \times D_{i-1}}$, we see that $\frac{\partial \ell}{\partial W_i} \in \mathbb{R}^{D_i \times D_{i-1}}$, which matches with W_i.

Since bias $\mathbf{b}_i$ appears additively, its derivative is simply multiplied element-wise by the derivation of the activation function:

$$\frac{\partial \ell}{\partial \mathbf{b}_i} = \frac{\partial \ell}{\partial \mathbf{h}_i} \odot \sigma'(W_i\mathbf{h}_{i-1} + \mathbf{b}_i) \tag{92}$$

Since $\frac{\partial \ell}{\partial \mathbf{h}_i} \in \mathbb{R}^{D_i \times 1}$ and $\sigma'(W_i\mathbf{h}_{i-1} + \mathbf{b}_i) \in \mathbb{R}^{D_i \times 1}$, their element-wise product and therefore $\frac{\partial \ell}{\partial \mathbf{b}_i} \in \mathbb{R}^{D_i \times 1}$.

3.3 [15pts] Derive the Kaiming initialization [3] in our context, *i.e.*, applying the same variance analysis technique we learned from Xavier initialization [4] to this deep MLP with ReLU activations.

For forward variance propagation, we can begin with assumptions based on weight initialization:

- The input to the first layer $\mathbf{h}_0 = \mathbf{x}$ is zero-mean and has variance $\mathbf{v}[\mathbf{h}_0] = v_0$
- The weights W_i are initialized with zero mean and variance $\mathbf{v}[\mathbf{h}_0] = \sigma_W^2$, and biases are ignored for simplicity.
- Each layer then computes $\mathbf{h}_i = \sigma(W_i\mathbf{h}_{i-1})$ where σ is the ReLU activation function.

Since ReLU introduces sparsity, it affects the variance propagation:

$$\mathbf{v}[\mathbf{h}_i] = \frac{1}{2} D_{i-1} \sigma_W^2 \mathbf{v}[h_{i-1}] \tag{93}$$

We can then expand recursively for all layers and set $v_L = v_0$ for stable activations:

$$v_L = \left(\frac{1}{2} D_0 \sigma_W^2\right)^L v_0 \tag{94}$$

$$\sigma_W^2 = \frac{2}{D_{i-1}} \tag{95}$$

For backward variance propagation, we can analyze the gradient variance:

$$\mathbf{v}\left[\frac{\partial \ell}{\partial \mathbf{h}_i}\right] = \frac{1}{2} D_i \sigma_W^2 \mathbf{v}\left[\frac{\partial \ell}{\partial \mathbf{h}_{i+1}}\right] \tag{96}$$

For stable gradient variance, we need $\sigma_W^2 = \frac{2}{D_i}$. To balance both forward and backwards variance conditions, we can use $\sigma_W^2 = \frac{2}{D_{i-1}}$. This means we can initialize weights as:

$$W_i \sim \mathcal{N}\left(0, \frac{2}{D_{i-1}}\right) \tag{97}$$