Deep Learning Winter 2025–2026

Homework 1 – Backpropagation

Solutions

Mahmoud Abade - 206773756

Feras Dwere 214225021

Question 1: Softmax & Cross-Entropy (15 Points)

Part (a): Derive $\frac{\partial L}{\partial z_i}$

Given:

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}, \quad i = 1, \dots, C \qquad \text{(Softmax)}$$

$$L = -\sum_{i=1}^{C} y_i \log p_i \qquad \text{(Cross-entropy loss)}$$

Derivation:
We use the chain rule:

$$\frac{\partial L}{\partial z_i} = \sum_{k=1}^{C} \frac{\partial L}{\partial p_k} \cdot \frac{\partial p_k}{\partial z_i} \tag{1}$$

Step 1: Compute $\frac{\partial L}{\partial p_k}$

$$\frac{\partial L}{\partial p_k} = \frac{\partial}{\partial p_k}\left[-\sum_{j=1}^{C} y_j \log p_j\right] = -\frac{y_k}{p_k} \tag{2}$$

Step 2: Use the softmax Jacobian (given in hint)

$$\frac{\partial p_k}{\partial z_i} = \begin{cases} p_i(1-p_i) & \text{if } k = i \\ -p_i p_k & \text{if } k \neq i \end{cases} \tag{3}$$

Step 3: Apply chain rule by splitting the sum

$$\begin{aligned}
\frac{\partial L}{\partial z_i} &= \frac{\partial L}{\partial p_i} \cdot \frac{\partial p_i}{\partial z_i} + \sum_{k \neq i} \frac{\partial L}{\partial p_k} \cdot \frac{\partial p_k}{\partial z_i} && (4) \\
&= \left(-\frac{y_i}{p_i}\right) \cdot p_i(1-p_i) + \sum_{k \neq i} \left(-\frac{y_k}{p_k}\right) \cdot (-p_i p_k) && (5) \\
&= -y_i(1-p_i) + \sum_{k \neq i} y_k p_i && (6) \\
&= -y_i + y_i p_i + p_i \sum_{k \neq i} y_k && (7) \\
&= -y_i + p_i \left(y_i + \sum_{k \neq i} y_k \right) && (8) \\
&= -y_i + p_i \sum_{k=1}^{C} y_k && (9)
\end{aligned}$$

Step 4: Since $\sum_{k=1}^{C} y_k = 1$ (one-hot encoding)

$$\boxed{\frac{\partial L}{\partial z_i} = p_i - y_i} \tag{10}$$

Part (b): Show why the derivative simplifies to $p_i - y_i$

Proof:

From part (a), we derived:

$$\frac{\partial L}{\partial z_i} = -y_i + p_i \sum_{k=1}^{C} y_k \tag{11}$$

The key insight is that **y is a one-hot vector**, meaning exactly one component equals 1 and all others equal 0. Therefore:

$$\sum_{k=1}^{C} y_k = 1 \tag{12}$$

Substituting this constraint:

$$\frac{\partial L}{\partial z_i} = -y_i + p_i \cdot 1 = p_i - y_i \tag{13}$$

Physical interpretation:

- If i is the **correct class**: $y_i = 1$, so gradient $= p_i - 1$ (negative, pushes z_i up)
- If i is an **incorrect class**: $y_i = 0$, so gradient $= p_i$ (positive, pushes z_i down)
- The gradient magnitude directly equals the prediction error

Part (c): Why does this provide strong learning signals?

The softmax + cross-entropy combination provides strong, non-vanishing gradients for the following reasons:

1. Linear gradient in logit space:

The gradient $\frac{\partial L}{\partial z_i} = p_i - y_i$ is linear in the prediction error. When the model is confidently wrong (e.g., $p_i \approx 1$ for an incorrect class or $p_i \approx 0$ for the correct class), the gradient magnitude is large (close to 1), providing a strong correction signal. The gradient is bounded in $[-1, 1]$ but never vanishes completely.

2. Cancellation of exponentials:

The exponential function in softmax and the logarithm in cross-entropy cancel out during backpropagation. This elegant cancellation prevents the vanishing gradient problem that would occur if we used softmax with mean squared error, where the gradient would include terms like $p_i(1 - p_i)$ that vanish as $p_i \to 0$ or $p_i \to 1$.

3. No saturation regions:

Unlike sigmoid or tanh activations that have flat regions where gradients vanish, this combination maintains informative gradients throughout. Even when the model is highly confident (correct or incorrect), the gradient magnitude appropriately reflects how much correction is needed: zero when correct, maximum (≈ 1) when confidently wrong.

Summary: The gradient magnitude directly reflects the prediction error, ensuring strong learning signals when the model needs correction most, while gracefully approaching zero when predictions are correct. This property enables efficient training even in the final stages when the model becomes confident.

Question 2: Batch Normalization (15 Points)

Given:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} z_i \qquad \text{(Batch mean)}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(z_i - \mu)^2 \qquad \text{(Batch variance)}$$

$$\hat{z}_i = \frac{z_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \qquad \text{(Normalization)}$$

$$y_i = \gamma \hat{z}_i + \beta \qquad \text{(Affine transform)}$$

where $\gamma, \beta \in \mathbb{R}$ are learnable parameters, $\epsilon > 0$ is a small constant, and $dy_i := \frac{\partial L}{\partial y_i}$ is the upstream gradient.

Part (a): Derive $\frac{\partial L}{\partial \gamma}$ and $\frac{\partial L}{\partial \beta}$

Gradient with respect to β:
Using the chain rule:

$$\frac{\partial L}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial L}{\partial y_i} \cdot \frac{\partial y_i}{\partial \beta} \tag{14}$$

Since $y_i = \gamma \hat{z}_i + \beta$, we have:

$$\frac{\partial y_i}{\partial \beta} = 1 \tag{15}$$

Therefore:

$$\boxed{\frac{\partial L}{\partial \beta} = \sum_{i=1}^{m} dy_i} \tag{16}$$

Gradient with respect to γ:
Similarly:

$$\frac{\partial L}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial L}{\partial y_i} \cdot \frac{\partial y_i}{\partial \gamma} \tag{17}$$

Since $y_i = \gamma \hat{z}_i + \beta$, we have:

$$\frac{\partial y_i}{\partial \gamma} = \hat{z}_i \tag{18}$$

Therefore:

$$\boxed{\frac{\partial L}{\partial \gamma} = \sum_{i=1}^{m} dy_i \cdot \hat{z}_i} \tag{19}$$

Part (b): Express $\frac{\partial L}{\partial z_i}$

Approach: Use chain rule through multiple paths: $z_i \to \hat{z}_i \to y_i$ (direct), $z_i \to \mu \to \hat{z}_j \to y_j$ (through mean), and $z_i \to \sigma^2 \to \hat{z}_j \to y_j$ (through variance).

Step 1: Define intermediate gradient

$$d\hat{z}_i := \frac{\partial L}{\partial \hat{z}_i} = \frac{\partial L}{\partial y_i} \cdot \frac{\partial y_i}{\partial \hat{z}_i} = dy_i \cdot \gamma \tag{20}$$

Step 2: Express total derivative

$$\frac{\partial L}{\partial z_i} = \frac{\partial L}{\partial \hat{z}_i} \cdot \frac{\partial \hat{z}_i}{\partial z_i} + \frac{\partial L}{\partial \mu} \cdot \frac{\partial \mu}{\partial z_i} + \frac{\partial L}{\partial \sigma^2} \cdot \frac{\partial \sigma^2}{\partial z_i} \tag{21}$$

Step 3: Compute each component
3a. Direct path:

$$\frac{\partial \hat{z}_i}{\partial z_i} = \frac{1}{\sqrt{\sigma^2 + \epsilon}} \tag{22}$$

3b. Path through μ:

$$\frac{\partial \mu}{\partial z_i} = \frac{1}{m} \tag{23}$$

$$\frac{\partial L}{\partial \mu} = \sum_{j=1}^{m} \frac{\partial L}{\partial \hat{z}_j} \cdot \frac{\partial \hat{z}_j}{\partial \mu} = \sum_{j=1}^{m} d\hat{z}_j \cdot \left(-\frac{1}{\sqrt{\sigma^2 + \epsilon}}\right) \tag{24}$$

$$= -\frac{1}{\sqrt{\sigma^2 + \epsilon}} \sum_{j=1}^{m} d\hat{z}_j \tag{25}$$

3c. Path through σ^2:

$$\frac{\partial \sigma^2}{\partial z_i} = \frac{2(z_i - \mu)}{m} \tag{26}$$

$$\frac{\partial L}{\partial \sigma^2} = \sum_{j=1}^{m} \frac{\partial L}{\partial \hat{z}_j} \cdot \frac{\partial \hat{z}_j}{\partial \sigma^2} \tag{27}$$

$$= \sum_{j=1}^{m} d\hat{z}_j \cdot \frac{-(z_j - \mu)}{2(\sigma^2 + \epsilon)^{3/2}} \tag{28}$$

$$= -\frac{1}{2(\sigma^2 + \epsilon)^{3/2}} \sum_{j=1}^{m} d\hat{z}_j (z_j - \mu) \tag{29}$$

$$= -\frac{1}{2(\sigma^2 + \epsilon)^{3/2}} \sum_{j=1}^{m} d\hat{z}_j \hat{z}_j \sqrt{\sigma^2 + \epsilon} \tag{30}$$

$$= -\frac{1}{2(\sigma^2 + \epsilon)} \sum_{j=1}^{m} d\hat{z}_j \hat{z}_j \tag{31}$$

Step 4: Combine all paths

$$\begin{aligned}\frac{\partial L}{\partial z_i} &= \frac{d\hat{z}_i}{\sqrt{\sigma^2+\epsilon}} + \left(-\frac{1}{\sqrt{\sigma^2+\epsilon}}\sum_{j=1}^{m} d\hat{z}_j\right)\cdot\frac{1}{m} \\ &\quad + \left(-\frac{1}{2(\sigma^2+\epsilon)}\sum_{j=1}^{m} d\hat{z}_j\hat{z}_j\right)\cdot\frac{2(z_i-\mu)}{m} \qquad (32)\\ &= \frac{d\hat{z}_i}{\sqrt{\sigma^2+\epsilon}} - \frac{1}{m\sqrt{\sigma^2+\epsilon}}\sum_{j=1}^{m} d\hat{z}_j - \frac{(z_i-\mu)}{m(\sigma^2+\epsilon)}\sum_{j=1}^{m} d\hat{z}_j\hat{z}_j \qquad (33)\\ &= \frac{d\hat{z}_i}{\sqrt{\sigma^2+\epsilon}} - \frac{1}{m\sqrt{\sigma^2+\epsilon}}\sum_{j=1}^{m} d\hat{z}_j - \frac{\hat{z}_i}{m}\sum_{j=1}^{m} d\hat{z}_j\hat{z}_j\cdot\frac{1}{\sqrt{\sigma^2+\epsilon}} \qquad (34)\end{aligned}$$

Step 5: Factor and substitute $d\hat{z}_j = \gamma \cdot dy_j$

$$\frac{\partial L}{\partial z_i} = \frac{1}{\sqrt{\sigma^2+\epsilon}}\left[d\hat{z}_i - \frac{1}{m}\sum_{j=1}^{m} d\hat{z}_j - \hat{z}_i\cdot\frac{1}{m}\sum_{j=1}^{m} d\hat{z}_j\hat{z}_j\right] \qquad (35)$$

Final Answer

$$\boxed{\frac{\partial L}{\partial z_i} = \frac{\gamma}{\sqrt{\sigma^2+\epsilon}}\left[dy_i - \frac{1}{m}\sum_{j=1}^{m} dy_j - \hat{z}_i\cdot\frac{1}{m}\sum_{j=1}^{m} dy_j\hat{z}_j\right]} \qquad (36)$$

Or equivalently:

$$\boxed{\frac{\partial L}{\partial z_i} = \frac{\gamma}{m\sqrt{\sigma^2+\epsilon}}\left[m\cdot dy_i - \sum_{j=1}^{m} dy_j - \hat{z}_i\sum_{j=1}^{m} dy_j\hat{z}_j\right]} \qquad (37)$$

Part (c): How does BN improve gradient flow?

Batch Normalization improves gradient flow and mitigates vanishing/exploding gradients through three key mechanisms:

1. Normalization of activations:

BN ensures that the inputs to each layer have approximately zero mean and unit variance ($\hat{z}_i$ has mean 0 and variance 1 over the batch). This prevents activations from becoming too large (exploding) or too small (vanishing) as they propagate through deep networks. By maintaining consistent activation scales across layers, gradients remain in a reasonable range during backpropagation, preventing the compounding effect of very large or very small gradients.

2. Reduced internal covariate shift:

By normalizing layer inputs, BN reduces the extent to which the distribution of each layer's inputs changes during training. This stabilization allows each layer to learn more independently without having to continuously adapt to shifting input distributions from previous layers. The reduced covariate shift prevents gradient pathologies that arise when distribution changes compound across many layers.

3. Gradient scaling and learnable parameters:

The gradient expression includes the factor $\frac{\gamma}{\sqrt{\sigma^2+\epsilon}}$, which provides automatic gradient rescaling. The learnable scale parameter γ allows the network to recover the representational power lost by normalization, while the normalization factor ensures gradients are neither too large nor too small. This enables the use of higher learning rates and accelerates convergence, especially in very deep networks (100+ layers).

Result: These properties collectively enable successful training of much deeper architectures that would otherwise suffer from severe vanishing or exploding gradient problems, making BN a crucial component in modern deep learning.

— **End of Solutions** —