Full name : Saad Benslimane
ID : 20228273
Email adress : saad.benslimane@umontreal.ca
February 16th, 2022

Instructions

- *For all questions, show your work !*
- *Use LaTeX when writing your answers. We recommend using this assignment latex code as a template. You may reuse most of the notation shorthands, equations and/or tables. See the assignment policy on the course website for more details.*
- *Submit your answers electronically via Gradescope.*
- ***TAs for this assignment are Matthew Scicluna and Akram Erraqabi.***

Question 1 (3-3-3-4-3-4-3)**.** Consider training a standard feed-forward neural network. For the purposes of this question we are interested in a single iteration of SGD on a single training example : $(\boldsymbol{x}, y)$. We denote $f(\boldsymbol{x}, \boldsymbol{\theta})$ as the output of the neural network with model parameters $\boldsymbol{\theta}$. Now let's say g is the output activation function and $a(\boldsymbol{x}, \boldsymbol{\theta})$ is the pre-activation network output such that $f(\boldsymbol{x}, \boldsymbol{\theta}) = g(a(\boldsymbol{x}, \boldsymbol{\theta}))$.

1.1 Assuming the network's goal is to do binary classification (with the detailed structure above), what would be an appropriate activation function for the output layer, i.e. what would be an appropriate function g ? *We will keep this choice of g for the rest of this exercise.*

Answer 1.1

The appropriate activation function g for a binary classification would be the sigmoid function :

$$g(\boldsymbol{z}) = 1/(1 + e^{-\boldsymbol{z}})$$

1.2 What does the output represent under this activation function ?

Answer 1.2

Under this activation function the output would be a value between 0 and 1, and since the sigmoid is a distribution function we can interpret it as probabilities.

1.3 Let $L_{CE}(f(\boldsymbol{x}, \boldsymbol{\theta}), y)$ be cross-entropy loss, express it as a function of $f(\boldsymbol{x}, \boldsymbol{\theta})$ and y.

Answer 1.3

The cross-entropy loss is :

$$L_{CE}(f(\boldsymbol{x}, \boldsymbol{\theta}), y) = -y \log(f(\boldsymbol{x}, \theta)) - (1-y)\log(1 - f(\boldsymbol{x}, \theta))$$

1.4 Compute the partial derivative $\frac{\partial L_{CE}(f(\boldsymbol{x}, \boldsymbol{\theta}), y)}{\partial a(\boldsymbol{x}, \boldsymbol{\theta})}$.

Answer 1.4

$$\begin{aligned}\frac{\partial L_{CE}(f(\boldsymbol{x}, \boldsymbol{\theta}), y)}{\partial a(\boldsymbol{x}, \boldsymbol{\theta})} &= -y\frac{\partial \log(f(\boldsymbol{x}, \theta))}{\partial a(\boldsymbol{x}, \boldsymbol{\theta})} - (1-y)\frac{\partial \log(1 - f(\boldsymbol{x}, \theta))}{\partial a(\boldsymbol{x}, \boldsymbol{\theta})} \\ &= \frac{-y}{f(\boldsymbol{x}, \theta)}\frac{\partial f(\boldsymbol{x}, \theta)}{\partial a(\boldsymbol{x}, \theta)} - \frac{1-y}{1 - f(\boldsymbol{x}, \theta)}\frac{\partial(1 - f(\boldsymbol{x}, \theta))}{\partial a(\boldsymbol{x}, \theta)}\end{aligned}$$

with :

$$\frac{\partial f(\boldsymbol{x},\theta)}{\partial a(\boldsymbol{x},\theta)} = \frac{e^{-a(\boldsymbol{x},\theta)}}{(1+e^{-a(\boldsymbol{x},\theta)})^2}$$
$$\frac{\partial(1-f(\boldsymbol{x},\theta)}{\partial a(\boldsymbol{x},\theta)} = \frac{-e^{-a(\boldsymbol{x},\theta)}}{(1+e^{-a(\boldsymbol{x},\theta)})^2}$$

The partial derivative become :

$$\begin{aligned}\frac{\partial L_{CE}(f(\boldsymbol{x},\boldsymbol{\theta}),y)}{\partial a(\boldsymbol{x},\boldsymbol{\theta})} &= -y(1-f(\boldsymbol{x},\theta)) + (1-y)f(\boldsymbol{x},\theta) \\ &= \begin{cases} -(1-f(\boldsymbol{x},\theta)) & \text{for } y=1 \\ f(\boldsymbol{x},\theta) & \text{for } y=0 \end{cases}\end{aligned}$$

1.5 Let $L_{MSE}(f(\boldsymbol{x},\boldsymbol{\theta}),y)$ be the mean-squared error, express it as a function of $f(\boldsymbol{x},\boldsymbol{\theta})$ and y (multiplicative factors can be ignored).

Answer 1.5

In this case the mean-squared error is :

$$L_{MSE}(f(\boldsymbol{x},\boldsymbol{\theta}),y) = (y-f(\boldsymbol{x},\theta))^2$$

1.6 Compute the partial derivative $\frac{\partial L_{MSE}(f(\boldsymbol{x},\boldsymbol{\theta}),y)}{\partial a(\boldsymbol{x},\boldsymbol{\theta})}$.

Answer 1.6

$$\begin{aligned}\frac{\partial L_{MSE}(f(\boldsymbol{x},\boldsymbol{\theta}),y)}{\partial a(\boldsymbol{x},\boldsymbol{\theta})} &= \frac{\partial}{\partial a(\boldsymbol{x},\theta)}[(y-f(\boldsymbol{x},\theta))^2] \\ &= -2(y-f(\boldsymbol{x},\theta))\frac{\partial f(\boldsymbol{x},\theta)}{\partial a(\boldsymbol{x},\theta)} \\ &= -2(y-f(\boldsymbol{x},\theta))\frac{e^{-a(\boldsymbol{x},\theta)}}{(1+e^{-a(\boldsymbol{x},\theta)})^2} \\ &= -2(y-f(\boldsymbol{x},\theta))(1-f(\boldsymbol{x},\theta))f(\boldsymbol{x},\theta) \\ &= \begin{cases} -2(1-f(\boldsymbol{x},\theta))^2 f(\boldsymbol{x},\theta) & \text{for } y=1 \\ 2(1-f(\boldsymbol{x},\theta))f(\boldsymbol{x},\theta)^2 & \text{for } y=0 \end{cases}\end{aligned}$$

1.7 Based on your answers to the above questions, what would be the more appropriate loss function for binary classification and why ?

Answer 1.7

from previous questions :
- The cross entropy loss is convex for binary classification because the second partial derivative equals the first partial derivative of the sigmoid function which has a clear minimum point and is guaranteed to minimize the cost function.
- The mean-squared error is not convex for binary classification because the first partial derivative have no clear minimum point, so it it's not guaranteed to minimize the cost function.

From that we can conclude that the more appropriate loss function for binary classification is cross entropy loss.

Question 2 (4-4-5-6)**.** Recall the definition of the softmax function : $S(\boldsymbol{x})_i = e^{\boldsymbol{x}_i}/\sum_j e^{\boldsymbol{x}_j}$.

2.1 Show that softmax is translation-invariant, that is : $S(\boldsymbol{x}+c) = S(\boldsymbol{x})$, where c is a scalar constant.

Answer 2.1

$$S(\boldsymbol{x}+c)_i = \frac{e^{\boldsymbol{x}_i+c}}{\sum_j e^{\boldsymbol{x}_j+c}} = \frac{e^c e^{\boldsymbol{x}_i}}{e^c \sum_j e^{\boldsymbol{x}_j}} = \frac{e^{\boldsymbol{x}_i}}{\sum_j e^{\boldsymbol{x}_j}} = S(\boldsymbol{x})_i$$

As we can see $S(\boldsymbol{x}+c) = S(\boldsymbol{x})$, so softmax is tramslation-invariant.

2.2 Let $\boldsymbol{x}$ be a 2-dimensional vector. One can represent a 2-class categorical probability using softmax $S(\boldsymbol{x})$. Show that $S(\boldsymbol{x})$ can be reparameterized using sigmoid function, i.e. $S(\boldsymbol{x}) = [\sigma(z), 1-\sigma(z)]^\top$ where z is a scalar function of $\boldsymbol{x}$.

Answer 2.2

In this case we have : $S(\boldsymbol{x}) = [\frac{e^{\boldsymbol{x}_1}}{e^{\boldsymbol{x}_1}+e^{\boldsymbol{x}_2}}, \frac{e^{\boldsymbol{x}_2}}{e^{\boldsymbol{x}_1}+e^{\boldsymbol{x}_2}}]^\top$
with :

$$\frac{e^{\boldsymbol{x}_1}}{e^{\boldsymbol{x}_1}+e^{\boldsymbol{x}_2}} = \frac{1}{1+e^{-(\boldsymbol{x}_1-\boldsymbol{x}_2)}} = \sigma(\boldsymbol{x}_1-\boldsymbol{x}_2)$$
$$\frac{e^{\boldsymbol{x}_2}}{e^{\boldsymbol{x}_1}+e^{\boldsymbol{x}_2}} = \frac{e^{-(\boldsymbol{x}_1-\boldsymbol{x}_2)}}{1+e^{-(\boldsymbol{x}_1-\boldsymbol{x}_2)}} = 1-\sigma(\boldsymbol{x}_1-\boldsymbol{x}_2)$$

finally :

$$S(\boldsymbol{x}) = [\sigma(z), 1-\sigma(z)]^\top$$

As we can see, $S(\boldsymbol{x})$ can be reparameterized using sigmoid function with $z = \boldsymbol{x}_1 - \boldsymbol{x}_2$.

2.3 Let $\boldsymbol{x}$ be a K-dimensional vector ($K \geq 2$). Show that $S(\boldsymbol{x})$ can be represented using $K-1$ parameters, i.e. $S(\boldsymbol{x}) = S([0, y_1, y_2, ..., y_{K-1}]^\top)$, where y_i is a scalar function of $\boldsymbol{x}$ for $i \in \{1, ..., K-1\}$.

Answer 2.3

As we showed in the first question, the softmax is translation-invariant : $S(\boldsymbol{x}+c)_i = S(\boldsymbol{x})_i$.
Using $c = -\boldsymbol{x}_0$, we have :

$$\begin{aligned} S(\boldsymbol{x})_i = S(\boldsymbol{x}-\boldsymbol{x}_0)_i &= S([0, \boldsymbol{x}_1-\boldsymbol{x}_0, \boldsymbol{x}_2-\boldsymbol{x}_0, ..., \boldsymbol{x}_{K-1}-\boldsymbol{x}_0]^\top) \\ &= S([0, \boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_{K-1}]^\top) \end{aligned}$$

With $\boldsymbol{y}_i = \boldsymbol{x}_i - \boldsymbol{x}_0$ for $\boldsymbol{x}$ for $i \in \{1, ..., K-1\}$, $S(\boldsymbol{x})$ can be represented using $K-1$ parameters.

2.4 Show that the Jacobian of the softmax function $J_{\text{softmax}}(\boldsymbol{x})$ can be expressed as : $\mathbf{Diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top$, where $\boldsymbol{p} = S(\boldsymbol{x})$.

Answer 2.4

On diagonal $i = j$:

$$\begin{aligned} \frac{\partial S(\boldsymbol{x})_i}{\partial \boldsymbol{x}_i} &= \frac{\sum_j e^{\boldsymbol{x}_j} e^{\boldsymbol{x}_i} - e^{\boldsymbol{x}_i} e^{\boldsymbol{x}_i}}{(\sum_j e^{\boldsymbol{x}_j})^2} \\ &= \frac{e^{\boldsymbol{x}_i}}{\sum_j e^{\boldsymbol{x}_j}} - \left(\frac{e^{\boldsymbol{x}_i}}{\sum_j e^{\boldsymbol{x}_j}}\right)^2 \\ &= S(\boldsymbol{x})_i - (S(\boldsymbol{x})_i)^2 \end{aligned}$$

Out diagonal $i \neq j$:

$$\begin{aligned}\frac{\partial S(\boldsymbol{x})_i}{\partial \boldsymbol{x}_j} &= \frac{-\sum_j e^{\boldsymbol{x}_j} e^{\boldsymbol{x}_i}}{(\sum_j e^{\boldsymbol{x}_j})^2} \\ &= \frac{-e^{\boldsymbol{x}_i}}{\sum_j e^{\boldsymbol{x}_j}} \frac{e^{\boldsymbol{x}_j}}{\sum_j e^{\boldsymbol{x}_j}} \\ &= -S(\boldsymbol{x})_i S(\boldsymbol{x})_j\end{aligned}$$

The Jacobian of the softmax function is :

$$\begin{aligned}J_{\text{softmax}}(\boldsymbol{x}) &= \begin{bmatrix} S(\boldsymbol{x})_0 - (S(\boldsymbol{x})_0)^2 & \cdots & -S(\boldsymbol{x})_0 S(\boldsymbol{x})_n \\ \vdots & \ddots & \vdots \\ -S(\boldsymbol{x})_0 S(\boldsymbol{x})_n & \cdots & S(\boldsymbol{x})_n - (S(\boldsymbol{x})_n)^2 \end{bmatrix} \\ &= \begin{bmatrix} S(\boldsymbol{x})_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & S(\boldsymbol{x})_n \end{bmatrix} - \begin{bmatrix} (S(\boldsymbol{x})_0)^2 & \cdots & S(\boldsymbol{x})_0 S(\boldsymbol{x})_n \\ \vdots & \ddots & \vdots \\ S(\boldsymbol{x})_0 S(\boldsymbol{x})_n & \cdots & (S(\boldsymbol{x})_n)^2 \end{bmatrix} \\ &= \mathbf{Diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top\end{aligned}$$

Question 3 (6)**.** Consider a 2-layer neural network $y : \mathbb{R}^D \to \mathbb{R}^K$ of the form :

$$y(x, \Theta, \sigma)_k = \sum_{j=1}^{M} \omega_{kj}^{(2)} \sigma \left(\sum_{i=1}^{D} \omega_{ji}^{(1)} x_i + \omega_{j0}^{(1)} \right) + \omega_{k0}^{(2)}$$

for $1 \le k \le K$, with parameters $\Theta = (\omega^{(1)}, \omega^{(2)})$ and logistic sigmoid activation function σ. Show that there exists an equivalent network of the same form, with parameters $\Theta' = (\tilde{\omega}^{(1)}, \tilde{\omega}^{(2)})$ and tanh activation function, such that $y(x, \Theta', \tanh) = y(x, \Theta, \sigma)$ for all $x \in \mathbb{R}^D$, and express Θ' as a function of Θ.

Answer 3

Let first find a relation between tanh and σ, we know that :

$$\tanh(\boldsymbol{x}) = \frac{e^{\boldsymbol{x}} - e^{-\boldsymbol{x}}}{e^{\boldsymbol{x}} + e^{-\boldsymbol{x}}} = 1 - 2\frac{e^{-\boldsymbol{x}}}{e^{\boldsymbol{x}} + e^{-\boldsymbol{x}}} = 1 - 2\frac{e^{-2\boldsymbol{x}}}{1 + e^{-2\boldsymbol{x}}} = 1 - 2\left(1 - \frac{1}{1 + e^{-2\boldsymbol{x}}}\right) = 2\sigma(2\boldsymbol{x}) - 1$$

Which means that :

$$\sigma(\boldsymbol{x}) = \frac{\tanh(x/2) + 1}{2}$$

for $1 \le k \le K, x \in \mathbb{R}^D$:

$$\begin{aligned}y(x, \Theta, \sigma)_k &= \sum_{j=1}^{M} \omega_{kj}^{(2)} \sigma \left(\sum_{i=1}^{D} \omega_{ji}^{(1)} x_i + \omega_{j0}^{(1)} \right) + \omega_{k0}^{(2)} \\ &= \sum_{j=1}^{M} \omega_{kj}^{(2)} \left[\frac{1}{2} \tanh \left(\sum_{i=1}^{D} \frac{1}{2} \omega_{ji}^{(1)} x_i + \frac{1}{2} \omega_{j0}^{(1)} \right) + \frac{1}{2} \right] + \omega_{k0}^{(2)} \\ &= \sum_{j=1}^{M} \frac{1}{2} \omega_{kj}^{(2)} \tanh \left(\sum_{i=1}^{D} \frac{1}{2} \omega_{ji}^{(1)} x_i + \frac{1}{2} \omega_{j0}^{(1)} \right) + \frac{1}{2} \sum_{j=1}^{M} \omega_{kj}^{(2)} + \omega_{k0}^{(2)}\end{aligned}$$

With :

$$\tilde{\omega}_{kj}^{(2)} = \frac{1}{2}\omega_{kj}^{(2)} \qquad \tilde{\omega}_{ji}^{(1)} = \frac{1}{2}\omega_{ji}^{(1)}$$

$$\tilde{\omega}_{k0}^{(2)} = \frac{1}{2}\sum_{j=1}^{M}\omega_{kj}^{(2)} + \omega_{k0}^{(2)} \qquad \tilde{\omega}_{j0}^{(1)} = \frac{1}{2}\omega_{j0}^{(1)}$$

There exists an equivalent network of same form :

$$y(x, \Theta', \tanh)_k = \sum_{j=1}^{M} \tilde{\omega}_{kj}^{(2)} \tanh\left(\sum_{i=1}^{D} \tilde{\omega}_{ji}^{(1)} x_i + \tilde{\omega}_{j0}^{(1)}\right) + \tilde{\omega}_{k0}^{(2)}$$

Question 4 (5-5-6)**.** Consider a convolutional neural network. Assume the input is a colorful image of size 128×128 in the RGB representation. The first layer convolves 32 8×8 kernels with the input, using a stride of 2 and a zero-padding of 3 (three zeros on each side). The second layer downsamples the output of the first layer with a 2×2 non-overlapping max pooling. The third layer convolves 64 3×3 kernels with a stride of 1 and a zero-padding of size 1 on each border.

4.1 What is the dimensionality of the output of the last layer, i.e. the number of scalars it contains ?
Answer 4.1
- For the first layer : $[(128, 128) + 2 \times (3, 3) - (8, 8)] \times \frac{1}{2} + (1, 1) = (64, 64)$
- For the second layer : $[(64, 64) - (2, 2)] \times \frac{1}{2} + (1, 1) = (32, 32)$
- For the last layer : $[(32, 32) + 2 \times (1, 1) - (3, 3)] \times 1 + (1, 1) = (32, 32)$

So the dimensionality of the output of the last layer is $(32, 32)$.

4.2 Not including the biases, how many parameters are needed for the last layer ?
Answer 4.2
The number of parameters are needed :
- For the first layer : $8 \times 8 \times 3 \times 32 = 6144$.
- For max pooling layer no parameter are needed.
- For the last layer : $3 \times 3 \times 64 \times 32 = 18432$.

4.3 Compute the *full*, *valid*, and *same* convolution (with kernel flipping) for the following 1D matrices : $\begin{bmatrix}1, 2, 3, 4\end{bmatrix} * \begin{bmatrix}1, 0, 2\end{bmatrix}$
Answer 4.3
- For $full$ convolution (with 2 zero padding), the output size would be : $[\frac{4+2\times2-3}{1}] + 1 = 6$

$$\begin{bmatrix}0, 0, 1, 2, 3, 4, 0, 0\end{bmatrix} * \begin{bmatrix}2, 0, 1\end{bmatrix} = \begin{bmatrix}1, 2, 5, 8, 6, 8\end{bmatrix}^\top$$

Details :

$$\begin{aligned} y[0] &= \left[0,0,1\right] * \left[2,0,1\right] = 1 \\ y[1] &= \left[0,1,2\right] * \left[2,0,1\right] = 2 \\ y[2] &= \left[1,2,3\right] * \left[2,0,1\right] = 5 \\ y[3] &= \left[1,2,3\right] * \left[2,0,1\right] = 8 \\ y[4] &= \left[2,3,4\right] * \left[2,0,1\right] = 6 \\ y[5] &= \left[4,0,0\right] * \left[2,0,1\right] = 8 \end{aligned}$$

The same way we got :
- For *valid* convolution (with 0 zero padding), the output size would be : $[\frac{4+2\times 0-3}{1}]+1=2$:

$$\left[1,2,3,4\right] * \left[2,0,1\right] = \left[5,8\right]^\top$$

- For *same* convolution (with 1 zero padding), the output size would be : $[\frac{4+2\times 1-3}{1}]+1=4$:

$$\left[0,1,2,3,4,0\right] * \left[2,0,1\right] = \left[2,5,8,6\right]^\top$$

Question 5 (2-5-6-2-5-6)**.** Let us use the notation $*$ and $\tilde{*}$ to denote the valid and full convolution operator **without kernel flipping**, respectively. The operations are defined as

$$\text{valid} : (\boldsymbol{x} * \boldsymbol{w})_n = \sum_{j=1}^{k} x_{n+j-1} w_j \tag{1}$$

$$\text{full} : (\boldsymbol{x} \mathbin{\tilde{*}} \boldsymbol{w})_n = \sum_{j=1}^{k} x_{n+j-k} w_j \ \ , \tag{2}$$

where k is the size of the kernel $\boldsymbol{w}$. As a convention, the value of a vector indexed "out-of-bound" is zero, e.g. if $\boldsymbol{x} \in \mathbb{R}^d$, then $x_i = 0$ for $i < 1$ and $i > d$. We define the flip operator which reverse the ordering of the components of a vector, i.e. $\text{flip}(\boldsymbol{w})_j = w_{k-j+1}$.

Consider a convolutional network with 1-D input $\boldsymbol{x} \in \mathbb{R}^d$. Its first and second convolutional layers have kernel $\boldsymbol{w}^1 \in \mathbb{R}^{k_1}$ and $\boldsymbol{w}^2 \in \mathbb{R}^{k_2}$, respectively. Assume $k_1 < d$ and $k_2 < d$. The network is specified as follows :

$$\begin{aligned} \boldsymbol{a}^1 &\leftarrow \boldsymbol{x} * \boldsymbol{w}^1 \text{ (valid convolution)} && (3) \\ \boldsymbol{h}^1 &\leftarrow \text{ReLU}(\boldsymbol{a}^1) && (4) \\ \boldsymbol{a}^2 &\leftarrow \boldsymbol{h}^1 * \boldsymbol{w}^2 \text{ (valid convolution)} && (5) \\ \boldsymbol{h}^2 &\leftarrow \text{ReLU}(\boldsymbol{a}^2) && (6) \\ &\ \ ... && (7) \\ L &\leftarrow ... && (8) \end{aligned}$$

where L is the loss.

5.1 What is the dimensionality of $\boldsymbol{a}^2$? Denote it by $|\boldsymbol{a}^2|$.

Answer 5.1

For the first *valid* convolution layer :

$$|\boldsymbol{a}^1| = [\frac{d + 2 \times 0 - k_1}{1}] + 1 = d - k_1 + 1$$

As ReLU won't change the dimensionality :

$$|\boldsymbol{h}^1| = |\boldsymbol{a}^1|$$

For the second *valid* convolution layer :

$$|\boldsymbol{a}^2| = [\frac{|\boldsymbol{h}^1| + 2 \times 0 - k_2}{1}] + 1 = d - k_1 - k_2 + 2$$

5.2 Derive $\frac{\partial a_i^2}{\partial h_n^1}$. Answer for all $i \in \{1, ..., |\boldsymbol{a}^2|\}$, given a particular n.

Answer 5.2

Using equation (1), we have : $a_n^2 = (\boldsymbol{h}^1 * \boldsymbol{w}^2)_n = \sum_{j=1}^{k_1} h_{n+j-1}^1 w_j^2$

$$\begin{aligned}\frac{\partial a_i^2}{\partial h_n^1} &= \frac{\partial}{\partial h_n^1}\left(\sum_{j=1}^{k_1} h_{i+j-1}^1 w_j^2\right) \\ &= \begin{cases} w_{n-i+1}^2 & \text{for } j = n-i+1 \\ 0 & \text{for } j \neq n-i+1 \end{cases}\end{aligned}$$

Note that this partial derivative amounts to zero if j not in the bounds of the kernel.
Final result is :

$$\begin{aligned}\frac{\partial a_i^2}{\partial h_n^1} &= \begin{cases} w_{n-i+1}^2 & \text{for } 1 \leq n-i+1 \leq k_2 \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} w_{n-i+1}^2 & \text{for } n-k_2+1 \leq i \leq n \\ 0 & \text{otherwise} \end{cases}\end{aligned}$$

5.3 Show that $\nabla_{\boldsymbol{h}^1} L = \nabla_{\boldsymbol{a}^2} L \tilde{*} \operatorname{flip}(\boldsymbol{w}^2)$ (full convolution). Start with

$$(\nabla_{\boldsymbol{h}^1} L)_n = \sum_{i=1}^{|\boldsymbol{a}^2|} (\nabla_{\boldsymbol{a}^2} L)_i \frac{\partial a_i^2}{\partial h_n^1}$$

Answer 5.3

From the previous question, the partial is non-zero when $i \in [n - k_2 + 1, n]$:

$$(\nabla_{\boldsymbol{h}^1} L)_n = \sum_{i=1}^{|\boldsymbol{a}^2|} (\nabla_{\boldsymbol{a}^2} L)_i \frac{\partial a_i^2}{\partial h_n^1} = \sum_{i=1}^{|\boldsymbol{a}^2|} (\nabla_{\boldsymbol{a}^2} L)_i w_{n-i+1}^2 = \sum_{i=n-k_2+1}^{n} (\nabla_{\boldsymbol{a}^2} L)_i w_{n-i+1}^2$$

Let's put $j = n - i + 1$, then j would run from $1 \to k_2$:

$$(\nabla_{\boldsymbol{h}^1} L)_n = \sum_{j=1}^{k_2} (\nabla_{\boldsymbol{a}^2} L)_{n-j+1} w_j^2$$

We know that $\text{flip}(\boldsymbol{w})_j = w_{k-j+1}$ that means $\text{flip}(\boldsymbol{w})_{k_2-j+1} = w_j$. Then :

$$(\nabla_{\boldsymbol{h}^1} L)_n = \sum_{j=1}^{k_2} (\nabla_{\boldsymbol{a}^2} L)_{n-j+1} \text{flip}(w^2)_{k_2-j+1}$$

Again, let's put $p = k_2 - j + 1$:

$$\begin{aligned}(\nabla_{\boldsymbol{h}^1} L)_n &= \sum_{p=1}^{k_2} (\nabla_{\boldsymbol{a}^2} L)_{n+p-k_2} \text{flip}(w^2)_p \\ &= (\nabla_{\boldsymbol{a}^2} L \,\tilde{*}\, \text{flip}(w^2))_n\end{aligned}$$

For the following, assume the convolutions in equations (3) and (5) are **full instead of valid**.

5.4 What is the dimensionality of $\boldsymbol{a}^2$? Denote it by $|\boldsymbol{a}^2|$.

<u>Answer 5.4</u>

For the first *valid* convolution layer :

$$|\boldsymbol{a}^1| = [\frac{d + 2 \times (k_1 - 1) - k_1}{1}] + 1 = d + k_1 - 1$$

As ReLU won't change the dimensionality :

$$|\boldsymbol{h}^1| = |\boldsymbol{a}^1|$$

For the second *valid* convolution layer :

$$|\boldsymbol{a}^2| = [\frac{|\boldsymbol{h}^1| + 2 \times (k_2 - 1) - k_2}{1}] + 1 = d + k_1 + k_2 - 2$$

5.5 Derive $\frac{\partial a_i^2}{\partial h_n^1}$. Answer for all $i \in \{1, ..., |\boldsymbol{a}^2|\}$, given a particular n.

<u>Answer 5.5</u>

Using equation (2), we have : $a_n^2 = (\boldsymbol{h}^1 \tilde{*} \boldsymbol{w}^2)_n = \sum_{j=1}^{k_2} h_{n+j-k_2}^1 w_j^2$

$$\begin{aligned}\frac{\partial a_i^2}{\partial h_n^1} &= \frac{\partial}{\partial h_n^1}\left(\sum_{j=1}^{k_2} h_{i+j-k_2}^1 w_j^2\right) \\ &= \begin{cases} w_{n-i+k_2}^2 & \text{for } j = n - i + k_2 \\ 0 & \text{for } j \neq n - i + k_2 \end{cases}\end{aligned}$$

Note that this partial derivative amounts to zero if j not in the bounds of the kernel.
Final result is :

$$\begin{aligned}\frac{\partial a_i^2}{\partial h_n^1} &= \begin{cases} w_{n-i+k_2}^2 & \text{for } 1 \leq n - i + k_2 \leq k_2 \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} w_{n-i+k_2}^2 & \text{for } n \leq i \leq n + k_2 - 1 \\ 0 & \text{otherwise} \end{cases}\end{aligned}$$

5.6 Show that $\nabla_{\boldsymbol{h}^1} L = \nabla_{\boldsymbol{a}^2} L * \text{flip}(\boldsymbol{w}^2)$ (valid convolution). Start with

$$(\nabla_{\boldsymbol{h}^1} L)_n = \sum_{i=1}^{|\boldsymbol{a}^2|} (\nabla_{\boldsymbol{a}^2} L)_i \frac{\partial a_i^2}{\partial h_n^1}$$

Answer 5.6

From the previous question, the partial is non-zero when $i \in [n, n + k_2 - 1]$:

$$(\nabla_{\boldsymbol{h}^1} L)_n = \sum_{i=1}^{|\boldsymbol{a}^2|} (\nabla_{\boldsymbol{a}^2} L)_i \frac{\partial a_i^2}{\partial h_n^1} = \sum_{i=1}^{|\boldsymbol{a}^2|} (\nabla_{\boldsymbol{a}^2} L)_i w_{n-i+k_2}^2 = \sum_{i=n}^{n+k_2-1} (\nabla_{\boldsymbol{a}^2} L)_i w_{n-i+k_2}^2$$

Let's put $j = n - i + k_2$, then j would run from $1 \to k_2$:

$$(\nabla_{\boldsymbol{h}^1} L)_n = \sum_{j=1}^{k_2} (\nabla_{\boldsymbol{a}^2} L)_{n-j+k_2} w_j^2$$

We know that $\text{flip}(\boldsymbol{w})_j = w_{k-j+1}$ that means $\text{flip}(\boldsymbol{w})_{k_2-j+1} = w_j$. Then :

$$(\nabla_{\boldsymbol{h}^1} L)_n = \sum_{j=1}^{k_2} (\nabla_{\boldsymbol{a}^2} L)_{n-j+k_2} \text{flip}(w^2)_{k_2-j+1}$$

Again, let's put $p = k_2 - j + 1$:

$$\begin{aligned}(\nabla_{\boldsymbol{h}^1} L)_n &= \sum_{p=1}^{k_2} (\nabla_{\boldsymbol{a}^2} L)_{n+p-1} \text{flip}(w^2)_p \\ &= (\nabla_{\boldsymbol{a}^2} L \ * \ \text{flip}(w^2))_n\end{aligned}$$