CSC311 Week 6 Practice Problems Backpropagation

Alice Gao

Contents

Notes: We denote the element-wise product of two vectors $\mathbf{a}$ and $\mathbf{b}$ as $\mathbf{a} \odot \mathbf{b}$, i.e.,

$$\mathbf{a} \odot \mathbf{b} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \odot \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} a_1 b_1 \\ a_2 b_2 \\ \vdots \\ a_n b_n \end{bmatrix}.$$

This operation is known as the *Hadamard product*.

1 Backpropagation

1.1 LO: Derive the non-vectorized backward pass equations given the forward pass equations for a single data point given a small neural network.

1. Consider the following computations. All the quantities are scalars.

$$\begin{aligned} z &= wx + b \\ y &= x + \mathsf{tanh}(z) \\ \mathcal{L} &= \frac{1}{2}(y - t)^2 \end{aligned}$$

(a) Draw a computation graph consisting of the quantities above.

(b) Compute the following derivatives using the backpropagation algorithm.

$$\frac{\partial \mathcal{L}}{\partial \mathcal{L}}, \quad \frac{\partial \mathcal{L}}{\partial y}, \quad \frac{\partial \mathcal{L}}{\partial z}, \quad \frac{\partial \mathcal{L}}{\partial w}, \quad \frac{\partial \mathcal{L}}{\partial b}, \quad \frac{\partial \mathcal{L}}{\partial x}$$

You can directly use the derivative below.

$$\frac{d}{dx}\tanh(x) = 1 - \tanh^2(x)$$

2. Suppose that we have a fully connected feedforward neural network with one hidden layer. The input layer has $D^{(0)}$ units, the hidden layer has $D^{(1)}$ units with ReLU activation, and the output layer has $D^{(2)}$ units with a softmax activation to produce a probability distribution over classes.

   We define the notation below to denote the components of the neural network.

   - $\mathbf{a}^{(0)} \in \mathbb{R}^{D^{(0)}}$ denotes the input vector.
   - $\mathbf{W}^{(1)} \in \mathbb{R}^{D^{(1)} \times D^{(0)}}$ and $\mathbf{b}^{(1)} \in \mathbb{R}^{D^{(1)}}$ denote the weights and biases from the input layer to the hidden layer.
   - $\mathbf{z}^{(1)}, \mathbf{a}^{(1)} \in \mathbb{R}^{D^{(1)}}$ denote the hidden layer pre-activations and activations, where the activation function is ReLU.
   - $\mathbf{W}^{(2)} \in \mathbb{R}^{D^{(2)} \times D^{(1)}}$ and $\mathbf{b}^{(2)} \in \mathbb{R}^{D^{(2)}}$ denote the weights and biases from the hidden layer to the output layer.
   - $\mathbf{z}^{(2)}, \mathbf{a}^{(2)} \in \mathbb{R}^{D^{(2)}}$ denote the output layer pre-activations and softmax probabilities.
   - $\mathbf{t} \in \mathbb{R}^{D^{(2)}}$ denotes the one-hot target vector.
   - $\mathcal{L}(\mathbf{a}^{(2)}, \mathbf{t})$ denotes the cross-entropy loss.

   The forward pass computations of this neural network for a single data point are given below.

$$
\begin{aligned}
z_j^{(1)} &= \sum_{i=1}^{D^{(0)}} W_{ji}^{(1)} a_i^{(0)} + b_j^{(1)}, && j = 1, \ldots, D^{(1)} \\
a_j^{(1)} &= \mathrm{ReLU}\left(z_j^{(1)}\right), && j = 1, \ldots, D^{(1)} \\
z_k^{(2)} &= \sum_{j=1}^{D^{(1)}} a_j^{(1)} W_{kj}^{(2)} + b_k^{(2)}, && k = 1, \ldots, D^{(2)} \\
a_k^{(2)} &= \frac{e^{z_k^{(2)}}}{\sum_{k'=1}^{D^{(2)}} e^{z_{k'}^{(2)}}}, && k = 1, \ldots, D^{(2)} \\
\mathcal{L}(\mathbf{a}^{(2)}, \mathbf{t}) &= -\sum_{k=1}^{D^{(2)}} t_k \log\left(a_k^{(2)}\right)
\end{aligned}
$$

In this question, we will derive the non-vectorized backward pass computations shown below.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial z_k^{(2)}} &= a_k^{(2)} - t_k, && k = 1, \ldots, D^{(2)} \\
\frac{\partial \mathcal{L}}{\partial W_{kj}^{(2)}} &= \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \, a_j^{(1)}, && k = 1, \ldots, D^{(2)},\ j = 1, \ldots, D^{(1)} \\
\frac{\partial \mathcal{L}}{\partial b_k^{(2)}} &= \frac{\partial \mathcal{L}}{\partial z_k^{(2)}}, && k = 1, \ldots, D^{(2)} \\
\frac{\partial \mathcal{L}}{\partial a_j^{(1)}} &= \sum_{k=1}^{D^{(2)}} \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \, W_{kj}^{(2)}, && j = 1, \ldots, D^{(1)} \\
\frac{\partial \mathcal{L}}{\partial z_j^{(1)}} &= \frac{\partial \mathcal{L}}{\partial a_j^{(1)}} \, \mathrm{ReLU}'(z_j^{(1)}), && j = 1, \ldots, D^{(1)} \\
\frac{\partial \mathcal{L}}{\partial W_{ji}^{(1)}} &= \frac{\partial \mathcal{L}}{\partial z_j^{(1)}} \, a_i^{(0)}, && j = 1, \ldots, D^{(1)},\ i = 1, \ldots, D^{(0)} \\
\frac{\partial \mathcal{L}}{\partial b_j^{(1)}} &= \frac{\partial \mathcal{L}}{\partial z_j^{(1)}}, && j = 1, \ldots, D^{(1)}
\end{aligned}$$

(a) Draw the vectorized computation graph.

(b) In the following steps, we will derive the non-vectorized partial derivatives for the backward pass. Show that

$$\frac{\partial \mathcal{L}}{\partial z_k^{(2)}} = a_k^{(2)} - t_k$$

(c) Show that

$$\frac{\partial \mathcal{L}}{\partial W_{kj}^{(2)}} = \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \cdot a_j^{(1)}$$

Sample Solutions: By the chain rule:

$$\frac{\partial \mathcal{L}}{\partial W_{kj}^{(2)}} = \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \cdot \frac{\partial z_k^{(2)}}{\partial W_{kj}^{(2)}}.$$

From the forward pass,

$$z_k^{(2)} = \sum_{j'=1}^{D^{(1)}} a_{j'}^{(1)} W_{kj'}^{(2)} + b_k^{(2)},$$

we see that:

$$\frac{\partial z_k^{(2)}}{\partial W_{kj}^{(2)}} = a_j^{(1)}.$$

Therefore:

$$\boxed{\frac{\partial \mathcal{L}}{\partial W_{kj}^{(2)}} = \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \cdot a_j^{(1)}}.$$

(d) Show that

$$\frac{\partial \mathcal{L}}{\partial b_k^{(2)}} = \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \cdot 1$$

(e) Show that

$$\frac{\partial \mathcal{L}}{\partial a_j^{(1)}} = \sum_{k=1}^{D^{(2)}} \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \cdot W_{kj}^{(2)}$$

Sample Solutions: Using the multi-variate chain rule:

$$\frac{\partial \mathcal{L}}{\partial a_j^{(1)}} = \sum_{k=1}^{D^{(2)}} \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \cdot \frac{\partial z_k^{(2)}}{\partial a_j^{(1)}}.$$

Here, k indexes output units $(1, \ldots, D^{(2)})$ and j indexes hidden layer units $(1, \ldots, D^{(1)})$.

From the forward pass:

$$z_k^{(2)} = \sum_{j'=1}^{D^{(1)}} a_{j'}^{(1)} W_{kj'}^{(2)} + b_k^{(2)},$$

we have:

$$\frac{\partial z_k^{(2)}}{\partial a_j^{(1)}} = W_{kj}^{(2)}.$$

Substituting into the chain rule expression:

$$\boxed{\frac{\partial \mathcal{L}}{\partial a_j^{(1)}} = \sum_{k=1}^{D^{(2)}} \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} W_{kj}^{(2)}}.$$

(f) Show that

$$\frac{\partial \mathcal{L}}{\partial z_j^{(1)}} = \frac{\partial \mathcal{L}}{\partial a_j^{(1)}} \operatorname{ReLU}'\big(z_j^{(1)}\big)$$

(g) Show that

$$\frac{\partial \mathcal{L}}{\partial W_{ji}^{(1)}} = \frac{\partial \mathcal{L}}{\partial z_j^{(1)}} \cdot a_i^{(0)}$$

(h) Show that

$$\frac{\partial \mathcal{L}}{\partial b_j^{(1)}} = \frac{\partial \mathcal{L}}{\partial z_j^{(1)}} \cdot 1$$

3. Consider the feedforward neural network shown below.

- The network has two input units, two hidden layers (each with ReLU activation), and a single output unit with a sigmoid activation.
- No units include bias terms.
- For each hidden unit h_j, the activation is

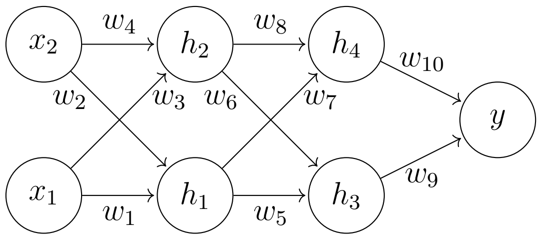
$$h_j = \text{ReLU}(m_j),$$

  where m_j is its total weighted input.
- The output unit y is computed as

$$y = \sigma(z),$$

  where z is the total weighted input to the output unit.

The goal is to minimize the loss $\mathcal{L}(y, t)$ using gradient descent, where t is the target label.



(a) Write down the forward pass computations. Specifically, write down a series of equations to calculate $m_1, h_1, m_2, h_2, m_3, h_3, m_4, h_4, z, y$.

Sample Solutions:

The forward pass computations are as follows.

$$\begin{aligned}
m_1 &= w_1 x_1 + w_2 x_2 \\
h_1 &= \mathrm{ReLU}(m_1) \\
m_2 &= w_3 x_1 + w_4 x_2 \\
h_2 &= \mathrm{ReLU}(m_2) \\
m_3 &= w_5 h_1 + w_6 h_2 \\
h_3 &= \mathrm{ReLU}(m_3) \\
m_4 &= w_7 h_1 + w_8 h_2 \\
h_4 &= \mathrm{ReLU}(m_4) \\
z &= w_9 h_3 + w_{10} h_4 \\
y &= \sigma(z) \\
\mathcal{L} &= \mathcal{L}(y, t)
\end{aligned}$$

(b) Write down the backward pass computations.

Sample Solutions:

The backward pass computations are as follows.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathcal{L}} &= 1 \\
\frac{\partial \mathcal{L}}{\partial y} &= \frac{\partial \mathcal{L}}{\partial \mathcal{L}} \mathcal{L}'(y, t) \\
\frac{\partial \mathcal{L}}{\partial z} &= \frac{\partial \mathcal{L}}{\partial y} \sigma'(z) \\
\frac{\partial \mathcal{L}}{\partial h_3} &= \frac{\partial \mathcal{L}}{\partial z} w_9 \\
\frac{\partial \mathcal{L}}{\partial h_4} &= \frac{\partial \mathcal{L}}{\partial z} w_{10} \\
\frac{\partial \mathcal{L}}{\partial m_3} &= \frac{\partial \mathcal{L}}{\partial h_3} \mathrm{ReLU}'(m_3) \\
\frac{\partial \mathcal{L}}{\partial m_4} &= \frac{\partial \mathcal{L}}{\partial h_4} \mathrm{ReLU}'(m_4)
\end{aligned}$$

(c) Your friend Ji-woo says that we should not compute $\frac{\partial \mathcal{L}}{\partial h_2}$ since the activation value h_2 is not a parameter that we learn. Explain why it is useful to compute $\frac{\partial \mathcal{L}}{\partial h_2}$ despite not learning h_2?

1.2 LO: Vectorize the forward pass equations for a small neural network.

1. **Vectorizing the forward pass for a single data point**

   Suppose that we have a fully connected feedforward neural network with one hidden layer. The input layer has $D^{(0)}$ units, the hidden layer has $D^{(1)}$ units with ReLU activation, and the output layer has $D^{(2)}$ units with a softmax activation to produce a probability distribution over classes.

   We define the notation below to denote the components of the neural network.

   - $D^{(i)}$ denotes the number of units in layer i.
   - $\mathbf{a}^{(0)} \in \mathbb{R}^{D^{(0)}}$ denotes the input vector.
   - $\mathbf{W}^{(1)} \in \mathbb{R}^{D^{(1)} \times D^{(0)}}$ and $\mathbf{b}^{(1)} \in \mathbb{R}^{D^{(1)}}$ denote the weights and biases from the input layer to the hidden layer.
   - $\mathbf{z}^{(1)}, \mathbf{a}^{(1)} \in \mathbb{R}^{D^{(1)}}$ denote the hidden layer pre-activations and activations, where the activation function is ReLU.
   - $\mathbf{W}^{(2)} \in \mathbb{R}^{D^{(2)} \times D^{(1)}}$ and $\mathbf{b}^{(2)} \in \mathbb{R}^{D^{(2)}}$ denote the weights and biases from the hidden layer to the output layer.
   - $\mathbf{z}^{(2)}, \mathbf{a}^{(2)} \in \mathbb{R}^{D^{(2)}}$ denote the output layer pre-activations and softmax probabilities.
   - $\mathbf{t} \in \mathbb{R}^{D^{(2)}}$ denotes the one-hot target vector.
   - $\mathbb{1}_{D^{(2)}}$ denotes a column vector of all ones of size $D^{(2)} \times 1$.
   - $\mathcal{L}(\mathbf{a}^{(2)}, \mathbf{t})$ denotes the cross-entropy loss.

   The non-vectorized forward pass computations are given below.

$$
\begin{aligned}
z_j^{(1)} &= \sum_{i=1}^{D^{(0)}} W_{ji}^{(1)} a_i^{(0)} + b_j^{(1)}, && j = 1, \ldots, D^{(1)} \\
a_j^{(1)} &= \mathrm{ReLU}\left(z_j^{(1)}\right), && j = 1, \ldots, D^{(1)} \\
z_k^{(2)} &= \sum_{j=1}^{D^{(1)}} a_j^{(1)} W_{kj}^{(2)} + b_k^{(2)}, && k = 1, \ldots, D^{(2)} \\
a_k^{(2)} &= \frac{e^{z_k^{(2)}}}{\sum_{k'=1}^{D^{(2)}} e^{z_{k'}^{(2)}}}, && k = 1, \ldots, D^{(2)} \\
\mathcal{L}(\mathbf{a}^{(2)}, \mathbf{t}) &= -\sum_{k=1}^{D^{(2)}} t_k \log\left(a_k^{(2)}\right)
\end{aligned}
$$

   Complete the task below to derive the vectorized computations step by step. Note that you may use $\exp$ and $\log$ as element-wise operations.

(a) Derived the vectorized expression for computing the pre-activation $\mathbf{z}^{(1)}$ as follows.

$$z_j^{(1)} = \sum_{i=1}^{D^{(0)}} W_{ji}^{(1)} a_i^{(0)} + b_j^{(1)}, \qquad j = 1, \ldots, D^{(1)}$$

(b) Derive the vectorized expression for the softmax function given below.

$$a_k^{(2)} = \frac{\exp\left(z_k^{(2)}\right)}{\sum_{k'=1}^{D^{(2)}} \exp\left(z_{k'}^{(2)}\right)}, \qquad k = 1, \ldots, D^{(2)}$$

Assume that $\exp$ is an element-wise operation and $\mathbb{1}_{D^{(2)}}$ is a column vector of ones of dimension $D^{(2)} \times 1$.

Sample Solutions: The solution is:

$$\mathbf{a}^{(2)} = \frac{\exp\left(\mathbf{z}^{(2)}\right)}{\mathbb{1}_{D^{(2)}}^{\top} \exp\left(\mathbf{z}^{(2)}\right)}$$

Since $\exp$ is an element-wise operation. We can apply $\exp$ to $z^{(2)}$ to obtain the corresponding vector $\exp\left(\mathbf{z}^{(2)}\right)$.

The denominator calculates a sum over the components of $\exp\left(\mathbf{z}^{(2)}\right)$. To calculate this summation, we can multiply the vector $\exp\left(\mathbf{z}^{(2)}\right)$ on the left by $(\mathbb{1}_{D^{(2)}})^{\top}$.

Therefore, the denominator

$$\sum_{k'=1}^{D^{(2)}} e^{z_{k'}^{(2)}}$$

becomes

$$\mathbb{1}_{D^{(2)}}^{\top} \exp\left(\mathbf{z}^{(2)}\right)$$

The final vectorized expression is

$$\mathbf{a}^{(2)} = \frac{\exp(\mathbf{z}^{(2)})}{\mathbb{1}_{D^{(2)}}^{\top} \exp(\mathbf{z}^{(2)})}.$$

(c) Derived the vectorized expression for computing the cross-entropy loss below.

$$\mathcal{L}(\mathbf{a}^{(2)}, \mathbf{t}) = -\sum_{k=1}^{D^{(2)}} t_k \log\left(a_k^{(2)}\right)$$

2. **Vectorizing the forward pass for a batch of data points**

Consider a fully connected feedforward neural network with one hidden layer. The hidden layer uses the ReLU activation function and the output layer uses the softmax activation. The loss function is the cross-entropy loss.

We define the following notation to denote the components of the neural network.

- N represents the batch size.
- $D^{(i)}$ represents the dimensionality (number of units) in layer i.
- $\mathbf{A}^{(i)}$ represents the $N \times D^{(i)}$ matrix of activations in layer i.
- $\mathbf{Z}^{(i)}$ represents the $N \times D^{(i)}$ matrix of pre-activations in layer i.
- $\mathbf{W}^{(i)}$ represents the $D^{(i)} \times D^{(i-1)}$ weight matrix between layer $i-1$ and layer i.
- $\mathbf{b}^{(i)}$ represents the $D^{(i)} \times 1$ bias vector for layer i.
- $\mathbf{T}$ represents the $N \times D^{(2)}$ matrix of target values.
- $\mathbb{1}_M$ represents a vector of all 1s of shape $M \times 1$ for any $M \in \mathbb{R}$.

In this question, our goal is to derive the vectorized forward pass computations for a batch of N data points.

Complete the tasks below to derive the vectorized backward pass computations step by step.

(a) Derive the vectorized expression for computing the pre-activations in layer 1 for a batch of data points, given the non-vectorized expression below.

$$Z_{nj}^{(1)} = \sum_{i=1}^{D^{(0)}} W_{ji}^{(1)} A_{ni}^{(0)} + b_j^{(1)}, \qquad n = 1, \ldots, N, j = 1, \ldots, D^{(1)}$$

(b) Derive the vectorized expression for computing the activations in layer 1 for a batch of data points given the non-vectorized expression below.

$$A_{nj}^{(1)} = \mathrm{ReLU}\left(Z_{nj}^{(1)}\right), \qquad n = 1, \ldots, N, j = 1, \ldots, D^{(1)}$$

(c) Derive the vectorized expression for computing the pre-activations in layer 2 for a batch of data points, given the non-vectorized expression below.

$$Z_{nk}^{(2)} = \sum_{j=1}^{D^{(1)}} W_{kj}^{(2)} A_{nj}^{(1)} + b_k^{(2)}, \qquad n = 1, \ldots, N, k = 1, \ldots, D^{(2)}$$

(d) Derived the vectorized expression for computing the activations in layer 2 for a batch of data points, given the non-vectorized expression below.

$$A_k^{(2)} = \frac{\exp\left(Z_k^{(2)}\right)}{\sum_{k'=1}^{D^{(2)}} \exp\left(Z_{k'}^{(2)}\right)}, \qquad k = 1, \ldots, D^{(2)}$$

(e) Derive the vectorized expression for computing the average loss for a batch of data points given the non-vectorized expression below.

$$\mathcal{L} = -\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{D^{(2)}} T_{nk} \log A_{nk}^{(2)}$$

1.3 LO: Vectorize the backward pass equations for a small neural network.

1. **Vectorizing the backward pass for a single data point**

   Consider a fully connected feedforward neural network with one hidden layer. The hidden layer uses the ReLU activation function and the output layer uses the softmax activation. The loss function is the cross-entropy loss.

   We define the notation below to denote the components of the neural network.

   - $D^{(i)}$ denotes the number of units in layer i.
   - $\mathbf{a}^{(0)} \in \mathbb{R}^{D^{(0)}}$ denotes the input vector.
   - $\mathbf{W}^{(1)} \in \mathbb{R}^{D^{(1)} \times D^{(0)}}$ and $\mathbf{b}^{(1)} \in \mathbb{R}^{D^{(1)}}$ denote the weights and biases from the input layer to the hidden layer.
   - $\mathbf{z}^{(1)}, \mathbf{a}^{(1)} \in \mathbb{R}^{D^{(1)}}$ denote the hidden layer pre-activations and activations, where the activation function is ReLU.
   - $\mathbf{W}^{(2)} \in \mathbb{R}^{D^{(2)} \times D^{(1)}}$ and $\mathbf{b}^{(2)} \in \mathbb{R}^{D^{(2)}}$ denote the weights and biases from the hidden layer to the output layer.
   - $\mathbf{z}^{(2)}, \mathbf{a}^{(2)} \in \mathbb{R}^{D^{(2)}}$ denote the output layer pre-activations and softmax probabilities.
   - $\mathbf{t} \in \mathbb{R}^{D^{(2)}}$ denotes the one-hot target vector.
   - $\mathcal{L}(\mathbf{a}^{(2)}, \mathbf{t})$ denotes the cross-entropy loss.

   The equations below are the backward pass computations of a neural network for a single data point. We will convert them to the corresponding vectorized equations.

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial z_k^{(2)}} &= a_k^{(2)} - t_k, && k = 1, \ldots, D^{(2)} \\
\frac{\partial \mathcal{L}}{\partial W_{kj}^{(2)}} &= \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \, a_j^{(1)}, && k = 1, \ldots, D^{(2)},\ j = 1, \ldots, D^{(1)} \\
\frac{\partial \mathcal{L}}{\partial b_k^{(2)}} &= \frac{\partial \mathcal{L}}{\partial z_k^{(2)}}, && k = 1, \ldots, D^{(2)} \\
\frac{\partial \mathcal{L}}{\partial a_j^{(1)}} &= \sum_{k=1}^{D^{(2)}} \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \, W_{kj}^{(2)}, && j = 1, \ldots, D^{(1)} \\
\frac{\partial \mathcal{L}}{\partial z_j^{(1)}} &= \frac{\partial \mathcal{L}}{\partial a_j^{(1)}} \, \mathrm{ReLU}'(z_j^{(1)}), && j = 1, \ldots, D^{(1)} \\
\frac{\partial \mathcal{L}}{\partial W_{ji}^{(1)}} &= \frac{\partial \mathcal{L}}{\partial z_j^{(1)}} \, a_i^{(0)}, && j = 1, \ldots, D^{(1)},\ i = 1, \ldots, D^{(0)} \\
\frac{\partial \mathcal{L}}{\partial b_j^{(1)}} &= \frac{\partial \mathcal{L}}{\partial z_j^{(1)}}, && j = 1, \ldots, D^{(1)}
\end{aligned}
$$

Complete the tasks below to derive the vectorized computations step by step.

(a) Derive the vectorized version of the expression below.

$$\frac{\partial \mathcal{L}}{\partial z_k^{(2)}} = a_k^{(2)} - t_k, \forall k = 1, \ldots, D^{(2)}$$

(b) Derive the vectorized version of the expression below.

$$\frac{\partial \mathcal{L}}{\partial W_{kj}^{(2)}} = \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} a_j^{(1)}, \qquad k = 1, \ldots, D^{(2)}, j = 1, \ldots, D^{(1)}$$

(c) Derive the vectorized version of the expression below.

$$\frac{\partial \mathcal{L}}{\partial b_k^{(2)}} = \frac{\partial \mathcal{L}}{\partial z_k^{(2)}}, \qquad k = 1, \ldots, D^{(2)}$$

(d) Derive the vectorized version of the expression below.

$$\frac{\partial \mathcal{L}}{\partial a_j^{(1)}} = \sum_{k=1}^{D^{(2)}} \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} W_{kj}^{(2)}, \qquad j = 1, \ldots, D^{(1)}$$

(e) Derive the vectorized version of the expression below.

$$\frac{\partial \mathcal{L}}{\partial z_j^{(1)}} = \frac{\partial \mathcal{L}}{\partial a_j^{(1)}} \operatorname{ReLU}'(z_j^{(1)}), \qquad j = 1, \ldots, D^{(1)}$$

Sample Solutions: The solution is

$$\nabla_{\mathbf{z}^{(1)}} \mathcal{L} = \nabla_{\mathbf{a}^{(1)}} \mathcal{L} \odot \operatorname{ReLU}'\Big(\mathbf{z}^{(1)}\Big)$$

Note that all three quantities are vectors of the same dimensions.

$$\nabla_{\mathbf{z}^{(1)}} \mathcal{L} \in \mathbb{R}^{D^1 \times 1}, \nabla_{\mathbf{a}^{(1)}} \mathcal{L} \in \mathbb{R}^{D^1 \times 1}, \operatorname{ReLU}'\Big(\mathbf{z}^{(1)}\Big) \in \mathbb{R}^{D^1 \times 1}$$

We are looking for an operation that starts with two vectors and produces a vector. This suggests that we should use the element-wise product.

$$\nabla_{\mathbf{z}^{(1)}} \mathcal{L} = \nabla_{\mathbf{a}^{(1)}} \mathcal{L} \odot \operatorname{ReLU}'\Big(\mathbf{z}^{(1)}\Big).$$

We can verify that the element-wise product works by writing the element-wise multiplication explicitly below.

$$\underbrace{\begin{bmatrix} \frac{\partial \mathcal{L}}{\partial a_1^{(1)}} \\ \frac{\partial \mathcal{L}}{\partial a_2^{(1)}} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial a_{D^{(1)}}^{(1)}} \end{bmatrix}}_{\nabla_{\mathbf{a}^{(1)}}\mathcal{L}} \odot \underbrace{\begin{bmatrix} \mathrm{ReLU}'(z_1^{(1)}) \\ \mathrm{ReLU}'(z_2^{(1)}) \\ \vdots \\ \mathrm{ReLU}'(z_{D^{(1)}}^{(1)}) \end{bmatrix}}_{\mathrm{ReLU}'(\mathbf{z}^{(1)})} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial a_1^{(1)}} \mathrm{ReLU}'(z_1^{(1)}) \\ \frac{\partial \mathcal{L}}{\partial a_2^{(1)}} \mathrm{ReLU}'(z_2^{(1)}) \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial a_{D^{(1)}}^{(1)}} \mathrm{ReLU}'(z_{D^{(1)}}^{(1)}) \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial z_1^{(1)}} \\ \frac{\partial \mathcal{L}}{\partial z_2^{(1)}} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial z_{D^{(1)}}^{(1)}} \end{bmatrix}.$$

Alternatively, we can convert the first vector to be a digonal matrix and use matrix-vector product instead.

$$\nabla_{\mathbf{z}^{(1)}}\mathcal{L} = \mathrm{diag}(\nabla_{\mathbf{a}^{(1)}}\mathcal{L})\, \mathrm{ReLU}'(\mathbf{z}^{(1)}).$$

(f) Derive the vectorized version of the expression below.

$$\frac{\partial \mathcal{L}}{\partial W_{ji}^{(1)}} = \frac{\partial \mathcal{L}}{\partial z_j^{(1)}}\, a_i^{(0)}, \quad j = 1, \ldots, D^{(1)}, i = 1, \ldots, D^{(0)}$$

(g) Derive the vectorized version of the expression below.

$$\frac{\partial \mathcal{L}}{\partial b_j^{(1)}} = \frac{\partial \mathcal{L}}{\partial z_j^{(1)}}, \qquad j = 1, \ldots, D^{(1)}$$

2. **Vectorizing the backward pass for a batch of data points**

   Consider a fully connected feedforward neural network with one hidden layer. The hidden layer uses the ReLU activation function and the output layer uses the softmax activation. The loss function is the cross-entropy loss.

   In this question, our goal is to derive the vectorized backward pass computations for a batch of N data points. To do this, we first define some notation below.

   - N is the batch size, $D^{(0)}$ is the number of input features, $D^{(1)}$ is the number of hidden units, and $D^{(2)}$ is the number of output classes.
   - $\mathbf{A}^{(0)} \in \mathbb{R}^{N \times D^{(0)}}$ is the data matrix.
   - $\mathbf{Z}^{(1)}, \mathbf{A}^{(1)} \in \mathbb{R}^{N \times D^{(1)}}$ are the pre- and post-ReLU hidden activations.
   - $\mathbf{Z}^{(2)}, \mathbf{A}^{(2)} \in \mathbb{R}^{N \times D^{(2)}}$ are the pre-softmax logits and post-softmax predictions.
   - $\mathbf{T} \in \mathbb{R}^{N \times D^{(2)}}$ is the matrix of target values.
   - $\mathbf{W}^{(1)} \in \mathbb{R}^{D^{(1)} \times D^{(0)}}$ and $\mathbf{W}^{(2)} \in \mathbb{R}^{D^{(2)} \times D^{(1)}}$ are the weight matrices.
   - $\mathbb{1}_N \in \mathbb{R}^{N \times 1}$ is a column vector of all ones for any $N \in \mathbb{R}$.

   Complete the tasks below to derive the vectorized backward pass computations for a batch of data points step by step.

   (a) Derive the vectorized expression for computing the gradient with respect to the pre-activations in layer 2 given the non-vectorized expression below.

   $$\frac{\partial \mathcal{L}}{\partial Z^{(2)}_{nk}} = \frac{1}{N}\left(A^{(2)}_{nk} - T_{nk}\right), \qquad k = 1, \ldots, D^{(2)}.$$

   (b) Derive the vectorized expression for computing the gradient with respect to the weights in layer 2 given the non-vectorized expression below.

   $$\frac{\partial \mathcal{L}}{\partial W^{(2)}_{kj}} = \sum_{n=1}^{N} \frac{\partial \mathcal{L}}{\partial Z^{(2)}_{nk}} A^{(1)}_{nj}, \qquad j = 1, \ldots, D^{(1)}, k = 1, \ldots, D^{(2)}$$

   (c) Derive the vectorized expression for computing the gradient with respect to the activations in layer 1 for a batch of data points given the non-vectorized expression below.

   $$\frac{\partial \mathcal{L}}{\partial A^{(1)}_{nj}} = \sum_{k=1}^{D^{(2)}} \frac{\partial \mathcal{L}}{\partial Z^{(2)}_{nk}} W^{(2)}_{kj}, \qquad n = 1, \ldots, N,\ j = 1, \ldots, D^{(1)}.$$

Sample Solutions: Starting from the given non-vectorized expression, for each example n and hidden unit j we have

$$\frac{\partial \mathcal{L}}{\partial A_{nj}^{(1)}} = \sum_{k=1}^{D^{(2)}} \frac{\partial \mathcal{L}}{\partial Z_{nk}^{(2)}} W_{kj}^{(2)}.$$

Now consider the matrices

$$\nabla_{\mathbf{Z}^{(2)}} \mathcal{L} \in \mathbb{R}^{N \times D^{(2)}}, \qquad \mathbf{W}^{(2)} \in \mathbb{R}^{D^{(2)} \times D^{(1)}}.$$

The (n, j)-entry of the matrix product $\nabla_{\mathbf{Z}^{(2)}} \mathcal{L}\, \mathbf{W}^{(2)}$ is

$$\left[\nabla_{\mathbf{Z}^{(2)}} \mathcal{L}\, \mathbf{W}^{(2)}\right]_{nj} = \sum_{k=1}^{D^{(2)}} \frac{\partial \mathcal{L}}{\partial Z_{nk}^{(2)}} W_{kj}^{(2)},$$

which matches the non-vectorized expression above.

Therefore, stacking the gradients over all $n = 1, \ldots, N$ gives the vectorized form

$$\boxed{\nabla_{\mathbf{A}^{(1)}} \mathcal{L} = \nabla_{\mathbf{Z}^{(2)}} \mathcal{L}\, \mathbf{W}^{(2)}}.$$

A dimension check confirms correctness:

$$[N \times D^{(2)}]\, [D^{(2)} \times D^{(1)}] = [N \times D^{(1)}].$$

(d) Derive the vectorized expression for computing the gradient with respect to the pre-activations in layer 1 for a batch of data points given the non-vectorized expression below.

$$\frac{\partial \mathcal{L}}{\partial Z_{nj}^{(1)}} = \frac{\partial \mathcal{L}}{\partial A_{nj}^{(1)}} \cdot \mathrm{ReLU}'\!\left(Z_{nj}^{(1)}\right), \qquad n = 1, \ldots, N,\ j = 1, \ldots, D^{(1)}.$$

(e) Derive the vectorized expression for computing the gradient with respect to the bias vector in layer 1 for a batch of data points given the non-vectorized expression below.

$$\frac{\partial \mathcal{L}}{\partial b_j^{(1)}} = \sum_{n=1}^{N} \frac{\partial \mathcal{L}}{\partial Z_{nj}^{(1)}}, \qquad j = 1, \ldots, D^{(1)}.$$

1.4 LO: Derive vectorized gradients for performing the backpropagation algorithm for small neural networks.

1. Consider the following neural network architecture, known as a **twin neural network**. This network computes the representations of two inputs and then compares these representations:

$$\begin{aligned}\mathbf{h}^{(a)} &= \Psi(\mathbf{x}^{(a)})\\ \mathbf{h}^{(b)} &= \Psi(\mathbf{x}^{(b)})\\ y &= \mathrm{dist}(\mathbf{h}^{(a)}, \mathbf{h}^{(b)})\end{aligned}$$

Here, Ψ computes the representation of an input, and dist compares two representations. In our case, both Ψ and dist are neural networks with the following architectures:

$$\begin{aligned}\Psi(\mathbf{x}) &= \mathrm{ReLU}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})\\ \mathrm{dist}(\mathbf{h}^{(a)}, \mathbf{h}^{(b)}) &= \sigma(\mathbf{W}^{(2)}(\mathbf{h}^{(a)} \odot \mathbf{h}^{(b)}) + b^{(2)})\end{aligned}$$

Where $\mathbf{h}^{(a)} \odot \mathbf{h}^{(b)}$ denotes element-wise multiplication:

$$(\mathbf{h}^{(a)} \odot \mathbf{h}^{(b)})_j = h_j^{(a)} \cdot h_j^{(b)}$$

To train this neural network, we will use the binary cross-entropy loss:

$$\mathcal{L} = -t \log y - (1-t)\log(1-y)$$

We will derive expressions for some gradients required for performing backpropagation in this neural network.

The forward pass computations of this neural network are given below.

$$\begin{aligned}\mathbf{m}^{(a)} &= \mathbf{W}^{(1)}\mathbf{x}^{(a)} + \mathbf{b}^{(1)}\\ \mathbf{h}^{(a)} &= \mathrm{ReLU}(\mathbf{m}^{(a)})\\ \mathbf{m}^{(b)} &= \mathbf{W}^{(1)}\mathbf{x}^{(b)} + \mathbf{b}^{(1)}\\ \mathbf{h}^{(b)} &= \mathrm{ReLU}(\mathbf{m}^{(b)})\\ \mathbf{h} &= \mathbf{h}^{(a)} \odot \mathbf{h}^{(b)}\\ z &= \mathbf{W}^{(2)}\mathbf{h} + b^{(2)}\\ y &= \sigma(z)\\ \mathcal{L} &= -t \log y - (1-t)\log(1-y)\end{aligned}$$

(a) Draw the computation graph for the twin neural network.

(b) Derive the expressions for computing $\nabla_{\mathbf{W}^2}\mathcal{L}$ and $\frac{\partial \mathcal{L}}{\partial b^{(2)}}$.

(c) Derive the expressions for computing $\nabla_{\mathbf{h}}\mathcal{L}$, $\nabla_{\mathbf{h}^{(a)}}\mathcal{L}$ and $\nabla_{\mathbf{h}^{(b)}}\mathcal{L}$.

(d) Derive the expressions for computing $\nabla_{\mathbf{W}^{(1)}}\mathcal{L}$ and $\nabla_{\mathbf{b}^{(1)}}\mathcal{L}$.

2. Consider a neural network defined by the computations below. The dimensions are given below.

   - Input $\mathbf{x} \in \mathbb{R}^{D\times 1}$
   - First hidden layer $\mathbf{h_1}$ has J units
   - Second hidden layer $\mathbf{h_2}$ has K units
   - The network aims to perform classification over M classes

   Forward pass equations

$$\begin{aligned}
&\mathbf{z_1} = \mathbf{W_1}\mathbf{x} + \mathbf{b_1}, && \mathbf{h_1} = \sigma(\mathbf{z_1}), && \\
&\mathbf{z_2} = \mathbf{W_2}\mathbf{h_1} + \mathbf{b_2}, && \mathbf{z_s} = \mathbf{W_s}\mathbf{x}, && \mathbf{h_2} = \mathrm{ReLU}(\mathbf{z_2}) + \mathbf{z_s}, \\
&\mathbf{z_3} = \mathbf{W_3}\mathbf{h_2} + \mathbf{b_3}, && \mathbf{y} = \mathrm{softmax}(\mathbf{z_3}), && \\
&\mathcal{L} = \mathcal{L}_{\mathrm{CE}}\big(\mathbf{y}, \mathbf{t}\big). && &&
\end{aligned}$$

   (a) Draw the vectorized computation graph.

   (b) Derive the gradients for layer 3 parameters.

$$\nabla_{\mathbf{z_3}}\mathcal{L}, \nabla_{\mathbf{W}_3}\mathcal{L}, \nabla_{\mathbf{b}_3}\mathcal{L}$$

   (c) Derive the gradients for layer 2 parameters below.

$$\nabla_{\mathbf{h_2}}\mathcal{L},\ \nabla_{\mathbf{z_2}}\mathcal{L},\ \nabla_{\mathbf{W}_2}\mathcal{L},\ \nabla_{\mathbf{b}_2}\mathcal{L}$$

   (d) Derive the gradients for layer 1 parameters below.

$$\nabla_{\mathbf{h_1}}\mathcal{L},\ \nabla_{\mathbf{z_1}}\mathcal{L},\ \nabla_{\mathbf{W}_1}\mathcal{L},\ \nabla_{\mathbf{b}_1}\mathcal{L}$$

   (e) Derive the following gradients.

$$\nabla_{\mathbf{z_s}}\mathcal{L},\ \nabla_{\mathbf{W}_s}\mathcal{L},\ \nabla_{\mathbf{x}}\mathcal{L}$$