For $\ell = 0, \cdots, L-1$, the weights between the ℓ-th layer and the $(\ell+1)$-th layer is represented by the matrix

$$\mathbf{W}^{(\ell)} = \begin{bmatrix} W_{1,1}^{(\ell)} & W_{1,2}^{(\ell)} & \cdots & W_{1,H}^{(\ell)} \\ W_{2,1}^{(\ell)} & W_{2,2}^{(\ell)} & \cdots & W_{2,H}^{(\ell)} \\ \vdots & \vdots & \ddots & \vdots \\ W_{H,1}^{(\ell)} & W_{H,2}^{(\ell)} & \cdots & W_{H,H}^{(\ell)} \end{bmatrix} \in \mathbb{R}^{H \times H}$$

such that $W_{i,j}^{(\ell)}$ represents the weight between the j-th node in the ℓ-th layer and the i-th node in the $(\ell+1)$-th layer and

$$\mathbf{h}^{(\ell+\mathbf{1})} = \mathbf{W}^{(\ell)}\mathbf{h}^{(\ell)}$$

After the final layer, the model computes some scalar loss $\mathcal{L} \in \mathbb{R}$.

6. We are interested in computing the gradient of the loss with respect to every parameter in this network. In this problem, we estimate the number of computations we need to take if we do it by hand as in the previous section.

a: What is the total number of weights in this network? Your answer should be in big O notation, in terms of H and L.

b: For the weights in the first layer, i.e., elements of $\mathbf{W}^{(\mathbf{0})}$, what is the total number of partial derivatives we need to compute to evaluate $\frac{\partial \mathcal{L}}{\partial W_{i,j}^{(0)}}$? You don't have to include any partial derivatives that you know will be 0. Your answer should be in big O notation, in terms of H and L.

c: If we compute each gradient separately, the total number of computations we need to make is approximately the product of (a) and (b). What relationship does this number have with the total number of weights in this network (your answer in (a))?

Hint: It might be helpful to look at the total number of edges in the network that exist in any path from $W_{i,j}^{(1)}$ to the final loss node $\mathcal{L}$.

7. The main motivation behind the backpropagation algorithm is that there is a lot of overlap in the intermediate computations when computing the gradients for different weights in the network. The backpropagation algorithm removes the redundancy in computing the same partial derivative by sequentially computing the gradients from the last layer to the first layer. Assume that you have already computed the gradient vector

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(\ell+1)}} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial h_1^{(\ell+1)}} \\ \frac{\partial \mathcal{L}}{\partial h_2^{(\ell+1)}} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial h_H^{(\ell+1)}} \end{bmatrix} \in \mathbb{R}^H$$

The backpropagation algorithm will compute the following gradients

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(\ell)}} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial W_{1,1}^{(\ell)}} & \frac{\partial \mathcal{L}}{\partial W_{1,2}^{(\ell)}} & \cdots & \frac{\partial \mathcal{L}}{\partial W_{1,H}^{(\ell)}} \\ \frac{\partial \mathcal{L}}{\partial W_{2,1}^{(\ell)}} & \frac{\partial \mathcal{L}}{\partial W_{1,2}^{(\ell)}} & \cdots & \frac{\partial \mathcal{L}}{\partial W_{2,H}^{(\ell)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{L}}{\partial W_{H,1}^{(\ell)}} & \frac{\partial \mathcal{L}}{\partial W_{H,2}^{(\ell)}} & \cdots & \frac{\partial \mathcal{L}}{\partial W_{H,H}^{(\ell)}} \end{bmatrix} \in \mathbb{R}^{H\times H} \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(\ell)}} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial h_1^{(\ell)}} \\ \frac{\partial \mathcal{L}}{\partial h_2^{(\ell)}} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial h_H^{(\ell)}} \end{bmatrix} \in \mathbb{R}^H$$

then recursively continue.

a: For any $1 \leq i, j, k \leq H$, rewrite the partial derivative $\frac{\partial h_i^{(\ell+1)}}{\partial W_{j,k}^{(\ell)}}$ using any of the known values. It may be helpful to use the Kronecker-Delta function

$$\delta_{i,k} = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases}$$

b: Using the result of (a), evaluate the gradient $\frac{\partial \mathcal{L}}{\partial W_{j,k}^{(\ell)}}$. How many partial derivatives do you need to additionally compute? You can exclude any partial derivative you know is 0.

c: Now combine the results of (b) in a matrix form

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(\ell)}} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial W_{1,1}^{(\ell)}} & \frac{\partial \mathcal{L}}{\partial W_{1,2}^{(\ell)}} & \cdots & \frac{\partial \mathcal{L}}{\partial W_{1,H}^{(\ell)}} \\ \frac{\partial \mathcal{L}}{\partial W_{2,1}^{(\ell)}} & \frac{\partial \mathcal{L}}{\partial W_{1,2}^{(\ell)}} & \cdots & \frac{\partial \mathcal{L}}{\partial W_{2,H}^{(\ell)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{L}}{\partial W_{H,1}^{(\ell)}} & \frac{\partial \mathcal{L}}{\partial W_{H,2}^{(\ell)}} & \cdots & \frac{\partial \mathcal{L}}{\partial W_{H,H}^{(\ell)}} \end{bmatrix}$$

and represent it as the outer product of two known vectors. Since we are combining the results of H^2 computations of (b), the total number of computations in this stage is H^2 times the number you computed in (b).

d: For any $1 \leq i, j \leq H$, evaluate the partial derivative $\frac{\partial h_i^{(\ell+1)}}{\partial h_j^{(\ell)}}$.

e: Using the result of (d), evaluate the gradient $\frac{\partial \mathcal{L}}{\partial h_j^{(\ell)}}$. How many partial derivatives do you need to additionally compute?

f: Now combine the results of (e) in a vector form

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(\ell)}} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial h_1^{(\ell)}} \\ \frac{\partial \mathcal{L}}{\partial h_2^{(\ell)}} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial h_H^{(\ell)}} \end{bmatrix}$$

and represent it as the product of a known matrix and a known vector. Since we are combining the results of H computations of (e), the total number of computations in this stage is H times the number you computed in (e).

g: Backpropagation terminates when we repeat this process for all L layers. Based on the result of (b) and (e), what is the total number of computations we need to do during one iteration of the backpropagation algorithm (known as the backward pass)? Your answer should be in big O notation, in terms of H and L. How does this number relate to the total number of weights in this network?

h (Extra Credit): What is the ratio of the number of computations for a forward pass versus the number of computations for a backward pass?