CS 316: Introduction to Deep Learning

Forward Propagation, Backward Propagation, and Computational Graphs

Week 6

Dr Abdul Samad

Lecture Outline

- Forward Propagation
- Computational Graphs
- Backward Propagation

Forward Propagation

- Refers to the calculation and storage of intermediate variables (including outputs) for a neural network in order from the input layer to the output layer.
- Let's begin with an example of a one-hidden layer MLP with weight decay.

Example - One Hidden Layer MLP

Assume the input is $\mathbf{x} \in \mathbb{R}^d$ and our hidden layer does not include a bias term.
The intermediate variable $\mathbf{z}$ is therefore equal to

$$\mathbf{z} = \mathbf{W}^{(1)}\mathbf{x}$$

where $\mathbf{W}^{(1)} \in \mathbb{R}^{h \times d}$ is the weight parameter in the hidden layer. After running the intermediate variable $\mathbf{z} \in \mathbb{R}^h$ through the activation function ϕ we obtain our hidden activation vector of length h.

$$\mathbf{h} = \phi(\mathbf{z})$$

The hidden layer output $\mathbf{h}$ is also an intermediate variable.Assuming that the parameters of the output layer only possess a weight of $\mathbf{W}^{(2)} \in \mathbb{R}^{q \times h}$, we can obtain an output layer variable with a vector of length q :

$$\mathbf{o} = \mathbf{W}^{(2)}\mathbf{h}$$

Example - One Hidden Layer MLP

Assuming that the loss function is l and the example label is y, we can then calculate the loss term for a single data example,

$$L = l(\mathbf{o}, y)$$

According to the definition of l_2 regularization that we will introduce later, given the hyperparameter λ, the regularization term is

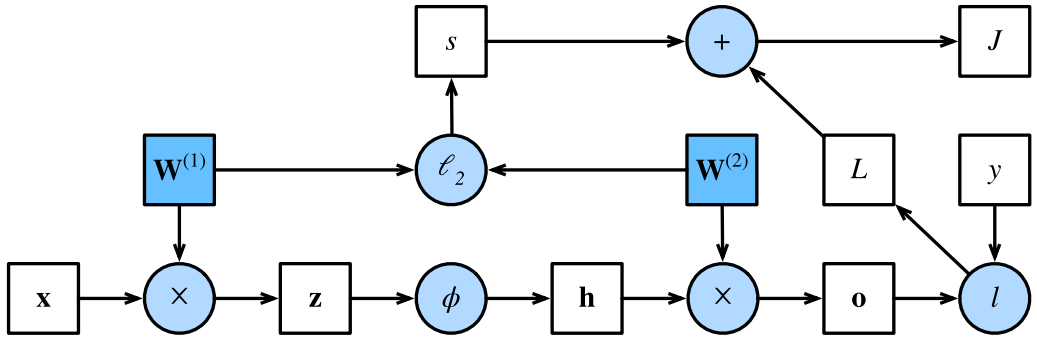
$$s = \frac{\lambda}{2}\left(\|\mathbf{W}^{(1)}\|_F^2 + \|\mathbf{W}^{(2)}\|_F^2\right)$$

where the Frobenius norm of the matrix is simply the $l2$ norm applied after flattening the matrix into a vector. Finally, the model's regularized loss on a given data example is:

$$J = L + s$$

Computational Graphs

- Computational graphs helps us visualize the dependencies of operators and variables within the calculation.
- The following is the computational graph of forward propagation of a single hidden layer multi layer perceptron that was discussed in the previous slides.



- The squares denotes the variables and circles denotes the operators. The lower left corner indicates the input and upper right corner indicates the output. Notice that the directions of the arrows (which illustrate data flow) are primarily rightward and upward.

Figure Source: Computational Graph of Forward Propagation

Backward Propagation

- Backpropagation refers to the method of calculating the gradient of the neural parameters.
- The method traverses the network in reverse order, from the output of the input layer, according to the chain rule from calculus.
- The algorithm stores any intermediate variables required while calculating the gradient with respect to some parameters.

Backward Propagation

Assume that we have the functions $\mathsf{Y} = f(\mathsf{X})$ and $\mathsf{Z} = g(\mathsf{Y})$, in which the input and output $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ are tensors of arbitrary shapes. By using the chain rule, we can compute the derivative of $\mathbf{Z}$ with respect to $\mathbf{X}$ via

$$\frac{\partial \mathsf{Z}}{\partial \mathsf{X}} = \text{prod}\left(\frac{\partial \mathsf{Z}}{\partial \mathsf{Y}}, \frac{\partial \mathsf{Y}}{\partial \mathsf{X}}\right).$$

Recall that the parameters of the simple network with one hidden layer which we discussed earlier are, $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$.

The objective of backpropagation is to calculate the gradients $\partial J/\partial \mathbf{W}^{(1)}$ and $\partial J/\partial \mathbf{W}^{(2)}$. To accomplish this, we apply the chain rule and calculate, in turn, the gradient of each intermediate variable and parameter.

The order of calculations is reversed relative to those performed in forward propagation since we need to start with the outcome of the computational graph and work our way towards the parameters.

Example - One Hidden Layer MLP

The first step is to calculate the gradients of the objective function $J = L+s$ with respect to the loss term L and the regularization term s

$$\frac{\partial J}{\partial L} = 1 \text{ and } \frac{\partial J}{\partial s} = 1.$$

Next, we compute the gradient of the objective function with respect to variable of the output layer $\mathbf{o}$ according to the chain rule:

$$\frac{\partial J}{\partial \mathbf{o}} = \text{prod}\left(\frac{\partial J}{\partial L}, \frac{\partial L}{\partial \mathbf{o}}\right) = \frac{\partial L}{\partial \mathbf{o}} \in \mathbb{R}^q.$$

Next, we calculate the gradients of the regularization term with respect to both parameters:

$$\frac{\partial s}{\partial \mathbf{W}^{(1)}} = \lambda \mathbf{W}^{(1)} \text{ and } \frac{\partial s}{\partial \mathbf{W}^{(2)}} = \lambda \mathbf{W}^{(2)}.$$

Example - One Hidden Layer MLP

Now we are able to calculate the gradient $\partial J/\partial \mathbf{W}^{(2)} \in \mathbb{R}^{q\times h}$of the model parameters closest to the output layer. Using the chain rule yields:

$$\frac{\partial J}{\partial \mathbf{W}^{(2)}} = \text{prod}\left(\frac{\partial J}{\partial \mathbf{o}}, \frac{\partial \mathbf{o}}{\partial \mathbf{W}^{(2)}}\right) + \text{prod}\left(\frac{\partial J}{\partial s}, \frac{\partial s}{\partial \mathbf{W}^{(2)}}\right) = \frac{\partial J}{\partial \mathbf{o}}\mathbf{h}^\top + \lambda \mathbf{W}^{(2)}.$$

To obtain the gradient with respect to $\mathbf{W}^{(1)}$ we need to continue backpropagation along the output layer to the hidden layer. The gradient with respect to the hidden layer output is given by

$$\frac{\partial J}{\partial \mathbf{h}} = \text{prod}\left(\frac{\partial J}{\partial \mathbf{o}}, \frac{\partial \mathbf{o}}{\partial \mathbf{h}}\right) = {\mathbf{W}^{(2)}}^\top \frac{\partial J}{\partial \mathbf{o}}.$$

Since the activation function ϕ applies elementwise, calculating the gradient $\partial J/\partial \mathbf{z} \in \mathbb{R}^h$ of the intermediate variable $\mathbf{z}$ requires that we use the elementwise multiplication operator, which we denote by $\odot$:

$$\frac{\partial J}{\partial \mathbf{z}} = \text{prod}\left(\frac{\partial J}{\partial \mathbf{h}}, \frac{\partial \mathbf{h}}{\partial \mathbf{z}}\right) = \frac{\partial J}{\partial \mathbf{h}} \odot \phi'\left(\mathbf{z}\right).$$

Finally, we can obtain the gradient $\partial J/\partial \mathbf{W}^{(1)} \in \mathbb{R}^{h\times d}$of the model parameters closest to the input layer. According to the chain rule, we get

$$\frac{\partial J}{\partial \mathbf{W}^{(1)}} = \text{prod}\left(\frac{\partial J}{\partial \mathbf{z}}, \frac{\partial \mathbf{z}}{\partial \mathbf{W}^{(1)}}\right) + \text{prod}\left(\frac{\partial J}{\partial s}, \frac{\partial s}{\partial \mathbf{W}^{(1)}}\right) = \frac{\partial J}{\partial \mathbf{z}}\mathbf{x}^\top + \lambda \mathbf{W}^{(1)}.$$

Forward and Backward Propagation

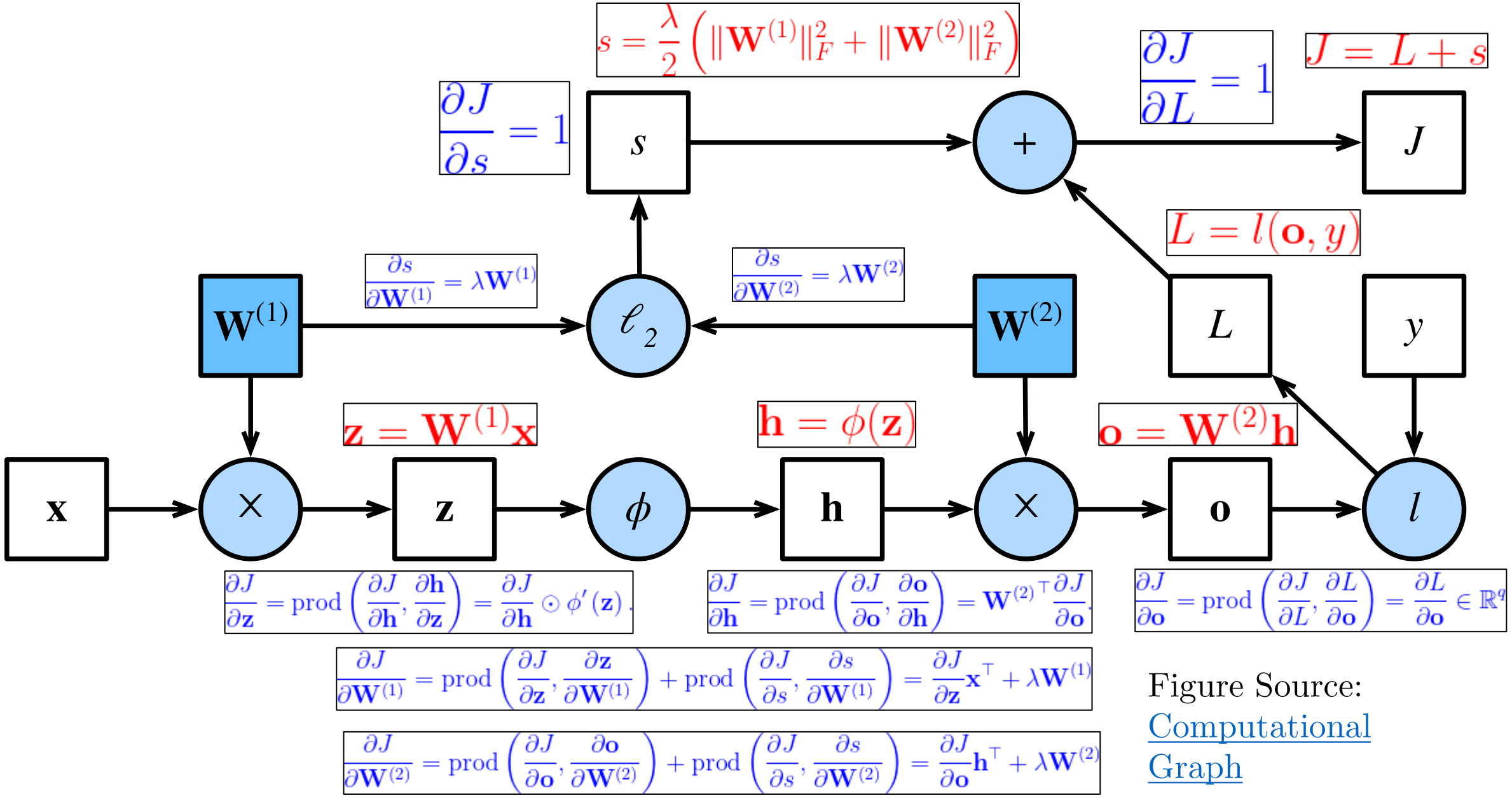


Figure Source: Computational Graph