4.2. The Backpropagation Algorithm

Lecture based on "Dive into Deep Learning" **http://D2L.AI** (Zhang et al., 2020)

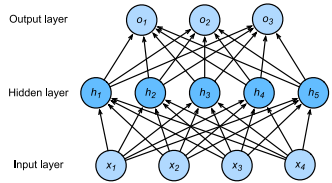
Prof. Dr. Christoph Lippert

Digital Health & Machine Learning

We'll discuss some of the details of **backward propagation** (more commonly called **backpropagation** or **backprop**).

Example

We will focus on a multilayer perceptron with a single hidden layer and ℓ_2 norm regularization.



Forward propagation refers to the *calculation* and *storage* of *intermediate variables (including outputs)* in the order from input layer to output layer.

- The input is $\mathbf{x} \in \mathbb{R}^d$ (no bias term).
- intermediate variable $\mathbf{z} \in \mathbb{R}^h$

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

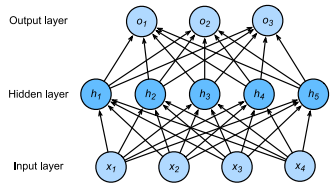
$\mathbf{W}^{(1)} \in \mathbb{R}^{h \times d}$ is the weight parameter of the hidden layer.

- After entering $\mathbf{z}$ into the activation function ϕ, we will obtain a hidden layer variable $\mathbf{h} \in \mathbb{R}^h$

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

- Using the weight parameter $\mathbf{W}^{(2)} \in \mathbb{R}^{q \times h}$, we compute an output layer $\mathbf{o} \in \mathbb{R}^q$:

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



- Given the loss function l and the label y, we calculate the loss L,

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

- Given the hyper-parameter λ, the ℓ_2 norm regularization term is

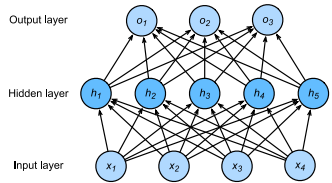
$$s = \frac{\lambda}{2} \left(\|\mathbf{W}^{(1)}\|_F^2 + \|\mathbf{W}^{(2)}\|_F^2 \right),$$

  where the Frobenius norm of the matrix is equivalent to the calculation of the L_2 norm after flattening the matrix to a vector.

- Finally, the model's regularized loss on a given data example is

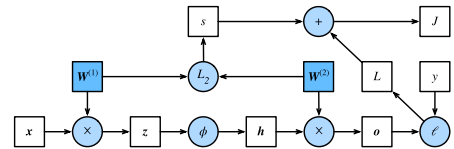
$$J = L + s.$$

  We refer to J as the objective function of a given data example.



Plotting **computational graphs** helps us visualize the dependencies of operators and variables within the calculation.

$$
\begin{aligned}
\mathbf{z} &= \mathbf{W}^{(1)}\mathbf{x} \\
\mathbf{h} &= \phi(\mathbf{z}). \\
\mathbf{o} &= \mathbf{W}^{(2)}\mathbf{h}. \\
L &= l(\mathbf{o}, y). \\
s &= \frac{\lambda}{2}\left(\|\mathbf{W}^{(1)}\|_F^2 + \|\mathbf{W}^{(2)}\|_F^2\right), \\
J &= L + s.
\end{aligned}
$$



Backpropagation refers to the method of calculating the gradient of neural network parameters.

- We calculate and store the intermediate variables of an objective function related to each layer.
- The gradient of the parameters are computer in the order of the output layer to the input layer according to the **chain rule** in calculus.

Let $y = f(\mathbf{u})$ and $\mathbf{u} = g(\mathbf{x})$, where $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{x} \in \mathbb{R}^n$, then the chain rule gives

$$\frac{\partial y}{\partial x_i} = \frac{\partial y}{\partial u_1}\frac{\partial u_1}{\partial x_i} + \frac{\partial y}{\partial u_2}\frac{\partial u_2}{\partial x_i} + \cdots + \frac{\partial y}{\partial u_m}\frac{\partial u_m}{\partial x_i}$$

for any $i = 1, 2, \ldots, n$.

Assume that we have functions

$\mathsf{Y} = f(\mathsf{X})$ and $\mathsf{Z} = g(\mathsf{Y}) = g \circ f(\mathsf{X})$, in which the input and the output $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$ are tensors of arbitrary shapes.

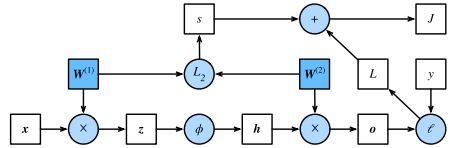
- By using the chain rule, we can compute the derivative of Z wrt. X via

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

  Here we use the prod operator to multiply its arguments after the necessary operations, such as transposition and swapping input positions have been carried out.

  - For vectors, this is simply matrix-matrix multiplication.
  - The operator prod hides all the notation overhead.

- The parameters of the simple network with one hidden layer are $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$.
- The objective of backpropagation is to calculate the gradients
  $\partial J/\partial \mathbf{W}^{(1)}$ and $\partial J/\partial \mathbf{W}^{(2)}$.
- We will apply the chain rule and calculate the gradient of each intermediate variable and parameter.
- We start with the outcome of the compute graph and work our way towards the parameters.



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

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

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

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

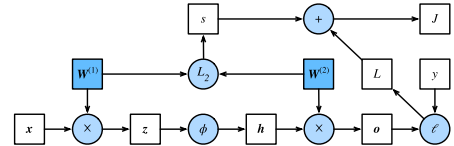
$$s = \frac{\lambda}{2}\left(\|\mathbf{W}^{(1)}\|_F^2 + \|\mathbf{W}^{(2)}\|_F^2\right),$$

$$J = L + s.$$

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

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

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



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

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

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

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

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

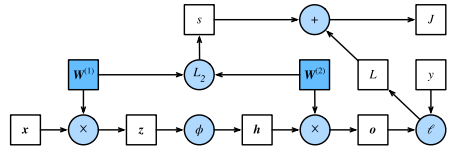
$$J = L + s.$$

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.

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



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

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

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

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

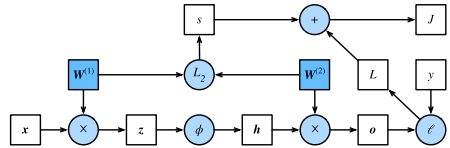
$$s = \frac{\lambda}{2}\left(\|\mathbf{W}^{(1)}\|_F^2 + \|\mathbf{W}^{(2)}\|_F^2\right),$$

$$J = L + s.$$

Now we are able 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:

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



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

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

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

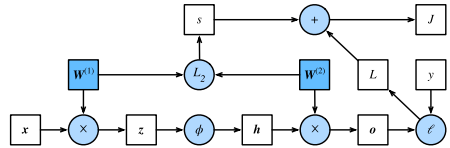
$$L = l(\mathbf{o}, y).$$

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

$$J = L + s.$$

The gradient with respect to the hidden layer's outputs $\partial J/\partial \mathbf{h} \in \mathbb{R}^h$ is given by

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



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

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

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

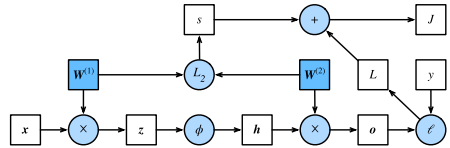
$$L = l(\mathbf{o}, y).$$

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

$$J = L + s.$$

Since the activation function ϕ applies element-wise, calculating the gradient $\partial J/\partial \mathbf{z} \in \mathbb{R}^h$ of the intermediate variable $\mathbf{z}$ requires the element-wise multiplication operator $\odot$.

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



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

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

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

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

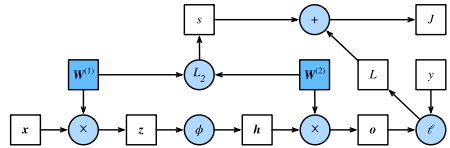
$$s = \frac{\lambda}{2}\left(\|\mathbf{W}^{(1)}\|_F^2 + \|\mathbf{W}^{(2)}\|_F^2\right),$$

$$J = L + s.$$

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

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



$$\mathbf{z} = \mathbf{W}^{(1)}\mathbf{x}$$
$$\mathbf{h} = \phi(\mathbf{z}).$$
$$\mathbf{o} = \mathbf{W}^{(2)}\mathbf{h}.$$
$$L = l(\mathbf{o}, y).$$
$$s = \frac{\lambda}{2}\left(\|\mathbf{W}^{(1)}\|_F^2 + \|\mathbf{W}^{(2)}\|_F^2\right),$$
$$J = L + s.$$

- When training networks, forward and backward propagation depend on each other.
- In particular, for forward propagation, we traverse through the compute graph in the direction of dependencies and compute all the variables on its path.
- These are then used for backpropagation where the compute order on the graph is reversed.
- One of the consequences is that we need to retain the intermediate values until backpropagation is complete.
- This is also one of the reasons why backpropagation requires significantly more memory than plain 'inference'—we end up computing tensors as gradients and need to retain all the intermediate variables to invoke the chain rule.
- Another reason is that we typically train with mini-batches containing more than one variable, thus more intermediate activations need to be stored.

Summary

- Forward propagation sequentially calculates and stores intermediate variables within the compute graph defined by the neural network. It proceeds from input to output layer.
- Back propagation sequentially calculates and stores the gradients of intermediate variables and parameters within the neural network in the reversed order.
- When training deep learning models, forward propagation and back propagation are interdependent.
- Training requires significantly more memory and storage.