Backpropagation
Overview

- Computation graphs
- Using the chain rule
- General backpropagation algorithm
- Toy examples of backward pass
- Matrix-vector calculations: ReLU, linear layer
Recall: Multi-layer neural networks

- The function computed by the network is a composition of the functions computed by individual layers (e.g., linear layers and nonlinearities):

- More precisely:
- What is the SGD update for the parameters $w_k$ of the kth layer?

$$w_k \leftarrow w_k - \eta \frac{\partial e}{\partial w_k}$$

- To train the network, we need to find the gradient of the error w.r.t. the parameters of each layer, $\frac{\partial e}{\partial w_k}$
Computation graph

\[ x \rightarrow f_1(x, w_1) \rightarrow h_1 \rightarrow f_2(h_1, w_2) \rightarrow h_2 \rightarrow \ldots \rightarrow f_K(h_{K-1}, w_K) \rightarrow h_K \rightarrow l(h_K, y) \rightarrow e \]
In calculus, the **chain rule** is a formula that expresses the derivative of the composition of two differentiable functions $f$ and $g$ in terms of the derivatives of $f$ and $g$. More precisely, if $h = f \circ g$ is the function such that $h(x) = f(g(x))$ for every $x$, then the chain rule is, in Lagrange's notation,

$$h'(x) = f'(g(x))g'(x).$$

or, equivalently,

$$h' = (f \circ g)' = (f' \circ g) \cdot g'.$$

The chain rule may also be expressed in Leibniz's notation. If a variable $z$ depends on the variable $y$, which itself depends on the variable $x$ (that is, $y$ and $z$ are dependent variables), then $z$ depends on $x$ as well, via the intermediate variable $y$. In this case, the chain rule is expressed as

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

Applying the chain rule

Let's start with $k = 1$

$e = l(f_1(x, w_1), y)$

Example: $e = (y - w_1^T x)^2$

$h_1 = f_1(x, w_1) = w_1^T x$

$e = l(h_1, y) = (y - h_1)^2$

$$
\frac{\partial e}{\partial w_1} = \frac{\partial h_1}{\partial w_1} = \frac{\partial e}{\partial h_1} =
$$
Applying the chain rule

\[ k = 2 \]

\[ e = l(f_2(f_1(x, w_1), w_2)) \]

Example: \[ e = -\log \left( \sigma(w_1^T x) \right) \] (assume \( y = 1 \))

\[ h_1 = f_1(x, w_1) = w_1^T x \]

\[ h_2 = f_2(h_1) = \sigma(h_1) \]

\[ e = l(h_2, 1) = -\log(h_2) \]
Applying the chain rule

\[ k = 2 \]

\[ e = l(f_2(f_1(x, w_1), w_2)) \]

Example: \( e = -\log(\sigma(w_1^T x)) \) (assume \( y = 1 \))

\[ h_1 = f_1(x, w_1) = w_1^T x \]
\[ h_2 = f_2(h_1) = \sigma(h_1) \]
\[ e = l(h_2, 1) = -\log(h_2) \]

\[ \frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial w_1} = \]

\[ \frac{\partial h_1}{\partial w_1} = \]
\[ \frac{\partial h_2}{\partial h_1} = \]
\[ \frac{\partial e}{\partial h_2} = \]
Chain rule: General case

\[ \frac{\partial e}{\partial w_k} = \frac{\partial e}{\partial h_k} \frac{\partial h_k}{\partial h_{k-1}} \cdots \frac{\partial h_k}{\partial h_{k-1}} \frac{\partial h_{k+1}}{\partial h_k} \]

Upstream gradient: \[ \frac{\partial e}{\partial h_k} \]

Local gradient: \[ \frac{\partial h_k}{\partial w_k} \]
Backpropagation summary

Parameter update:
\[
\frac{\partial e}{\partial w_k} = \frac{\partial e}{\partial h_k} \frac{\partial h_k}{\partial w_k}
\]

Upstream gradient:
\[
\frac{\partial e}{\partial h_k}
\]

Downstream gradient:
\[
\frac{\partial e}{\partial h_{k-1}} = \frac{\partial e}{\partial h_k} \frac{\partial h_k}{\partial h_{k-1}}
\]
What about more general computation graphs?

ResNet  

ResNeXt

Figure source
What about more general computation graphs?

Parameter update:
\[
\frac{\partial e}{\partial w_k} = \frac{\partial e}{\partial h_k} \frac{\partial h_k}{\partial w_k}
\]

Upstream gradient:
\[
\frac{\partial e}{\partial h_{k,1}}
\]

Downstream gradient:
\[
\frac{\partial e}{\partial h_{k-1}} = \frac{\partial e}{\partial h_k} \frac{\partial h_k}{\partial h_{k-1}}
\]

Local gradient

Local gradient

+ Gradients add at branches

\[
\frac{\partial e}{\partial h_{k,2}}
\]
Overview

• Computation graphs
• Using the chain rule
• General backprop algorithm
• Toy examples of backward pass
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w^{(0)}x^{(0)} + w^{(1)}x^{(1)} + w^{(2)})]} \]

Source: Stanford 231n
A detailed example

\[
f(x, w) = \frac{1}{1 + \exp[-(w^{(0)}x^{(0)} + w^{(1)}x^{(1)} + w^{(2)})]}
\]

\[
(1/x)' = -1/x^2
\]

\[
- \frac{1}{1.37^2} \times 1 = -0.53
\]

Source: Stanford 231n
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w^{(0)}x^{(0)} + w^{(1)}x^{(1)} + w^{(2)})]} \]
A detailed example

\[
f(x, w) = \frac{1}{1 + \exp[-(w^{(0)}x^{(0)} + w^{(1)}x^{(1)} + w^{(2)})]} \]

\[
\exp(-1) \times (-0.53) = -0.20
\]

Source: Stanford 231n
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w^{(0)}x^{(0)} + w^{(1)}x^{(1)} + w^{(2)})]} \]
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w^{(0)}x^{(0)} + w^{(1)}x^{(1)} + w^{(2)})]} \]
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w^{(0)}x^{(0)} + w^{(1)}x^{(1)} + w^{(2)})]} \]

Source: Stanford 231n
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w^0 x^0 + w^1 x^1 + w^2)]} \]

Source: Stanford 231n
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w^{(0)}x^{(0)} + w^{(1)}x^{(1)} + w^{(2)})]} \]
A detailed example

\[
f(x, w) = \frac{1}{1 + \exp[-(w^{(0)} x^{(0)} + w^{(1)} x^{(1)} + w^{(2)})]}\]

Source: Stanford 231n
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w^{(0)}x^{(0)} + w^{(1)}x^{(1)} + w^{(2)})]} \]

Source: Stanford 231n
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w^{(0)}x^{(0)} + w^{(1)}x^{(1)} + w^{(2)})]} \]
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w^{(0)}x^{(0)} + w^{(1)}x^{(1)} + w^{(2)})]} \]
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w^{(0)}x^{(0)} + w^{(1)}x^{(1)} + w^{(2)})]} \]

Can simplify computation graph

Sigmoid gate \( \sigma(x) \)
\[
\sigma'(x) = \sigma(x)(1 - \sigma(x)) \\
\sigma(1)(1 - \sigma(1)) = 0.73 \times (1 - 0.73) = 0.20
\]

Source: Stanford 231n
Another example

Source: Stanford 231n
Another example

Add gate: "gradient distributor"

Source: Stanford 231n
Another example

Add gate: “gradient distributor”
Multiply gate: “gradient switcher”

Source: Stanford 231n
Another example

Add gate: “gradient distributor”
Multiply gate: “gradient switcher”
Max gate: “gradient router”

Source: Stanford 231n
Overview: Backpropagation

- Computation graphs
- Using the chain rule
- General backprop algorithm
- Toy examples of backward pass
- Matrix-vector calculations: ReLU, linear layer
Backpropagation summary

Parameter update:
\[
\frac{\partial e}{\partial w_k} = \frac{\partial e}{\partial h_k} \frac{\partial h_k}{\partial w_k}
\]

Upstream gradient:
\[
\frac{\partial e}{\partial h_k}
\]

Downstream gradient:
\[
\frac{\partial e}{\partial h_{k-1}} = \frac{\partial e}{\partial h_k} \frac{\partial h_k}{\partial h_{k-1}}
\]

Local gradient

\( w_k \)

\( h_{k-1} \)

\( h_k \)

\( f_k \)
Dealing with vectors

\[ \frac{\partial z}{\partial x} = \begin{pmatrix} \frac{\partial z^{(1)}}{\partial x^{(1)}} & \cdots & \frac{\partial z^{(1)}}{\partial x^{(M)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial z^{(N)}}{\partial x^{(1)}} & \cdots & \frac{\partial z^{(N)}}{\partial x^{(M)}} \end{pmatrix} \]

**Jacobian:** row indices correspond to outputs, column indices correspond to inputs. The \( i, j \)th element of the Jacobian is the partial derivative of the \( i \)th output w.r.t. \( j \)th input.
Simple case: Elementwise operation
Simple case: Elementwise operation (ReLU layer)

The Jacobian for an elementwise function looks like:

\[
\frac{\partial z}{\partial x} = \begin{pmatrix}
\frac{\partial z^{(1)}}{\partial x^{(1)}} & \cdots & \frac{\partial z^{(1)}}{\partial x^{(M)}} \\
\vdots & \ddots & \vdots \\
\frac{\partial z^{(M)}}{\partial x^{(1)}} & \cdots & \frac{\partial z^{(M)}}{\partial x^{(M)}}
\end{pmatrix}
\]

What does the Jacobian for an elementwise function look like?
Simple case: Elementwise operation (ReLU layer)

\[
\frac{\partial z}{\partial x} = \begin{pmatrix}
\frac{\partial z^{(1)}}{\partial x^{(1)}} & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & \frac{\partial z^{(M)}}{\partial x^{(M)}}
\end{pmatrix}
\]

What does the Jacobian for an elementwise function look like?
Simple case: Elementwise operation (ReLU layer)

\[
\frac{\partial z}{\partial x} = \begin{pmatrix}
\mathbb{1}[x^{(1)} > 0] & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & \mathbb{1}[x^{(M)} > 0]
\end{pmatrix}
\]

What happens if some \( x^{(i)} \) is always negative?

This is known as the “dead ReLU” problem.
Matrix-vector multiplication (linear layer)

\[
f(x, W) = xW
\]

\[
\frac{\partial e}{\partial W} = \frac{\partial e}{\partial z} \frac{\partial z}{\partial W}
\]

\[
M \times N \quad 1 \times N \quad N \times (M \times N)
\]

\[
\frac{\partial z}{\partial W}
\]

\[
N \times (M \times N)
\]

\[
\frac{\partial e}{\partial z}
\]

\[
1 \times N
\]

\[
\frac{\partial e}{\partial x}
\]

\[
1 \times M \quad 1 \times N \quad N \times M
\]
Matrix-vector multiplication (linear layer)

\[
(z^{(1)} \ldots z^{(N)}) = (x^{(1)} \ldots x^{(M)}) \begin{pmatrix} W^{(11)} & \cdots & W^{(1N)} \\ \vdots & \ddots & \vdots \\ W^{(M1)} & \cdots & W^{(MN)} \end{pmatrix} = \sum_{i=1}^{M} x^{(i)} W^{(ij)}
\]

Want:

\[
\frac{\partial e}{\partial x} = \begin{bmatrix}
\frac{\partial e}{\partial z} \\
\frac{\partial z}{\partial x}
\end{bmatrix}
\]

\[
\frac{\partial z^{(j)}}{\partial x^{(i)}} = j\text{th row, } i\text{th column of Jacobian}
\]

\[
\frac{\partial z}{\partial x} = W^T
\]
Matrix-vector multiplication (linear layer)

\[
(z^{(1)} \ldots z^{(N)}) = (x^{(1)} \ldots x^{(M)}) \begin{pmatrix} W^{(11)} & \cdots & W^{(1N)} \\ \vdots & \ddots & \vdots \\ W^{(M1)} & \cdots & W^{(MN)} \end{pmatrix} z(j) = \sum_{i=1}^{M} x^{(i)} W^{(ij)}
\]

Want: \[
\frac{\partial e}{\partial x} = \frac{\partial e}{\partial z} \frac{\partial z}{\partial x}
\]

\[
1 \times M \quad 1 \times N \quad N \times M
\]

\[
\frac{\partial z^{(j)}}{\partial x^{(i)}} = W^{(ij)} \quad \text{jth row, ith column of Jacobian}
\]

\[
\frac{\partial z}{\partial x} = W^T
\]

\[
\frac{\partial e}{\partial x} = \frac{\partial e}{\partial z} \frac{\partial z}{\partial x} = \frac{\partial e}{\partial z} W^T
\]
Matrix-vector multiplication (linear layer)

\[ (z^{(1)} \ldots z^{(N)}) = (x^{(1)} \ldots x^{(M)}) \begin{pmatrix} W^{(11)} & \ldots & W^{(1N)} \\ \vdots & \ddots & \vdots \\ W^{(M1)} & \ldots & W^{(MN)} \end{pmatrix} \]

\[ z(j) = \sum_{i=1}^{M} x^{(i)}W^{(ij)} \]

Want: \[ \frac{\partial e}{\partial W} = \left[ \frac{\partial e}{\partial z} \frac{\partial z}{\partial W} \right] \]

\[ \frac{\partial z^{(k)}}{\partial W^{(ij)}} \]

\[ z^{(k)} \text{ depends only on } k\text{th column of } W \]
Matrix-vector multiplication (linear layer)

\[
(z^{(1)} \ldots z^{(N)}) = (x^{(1)} \ldots x^{(M)}) \begin{pmatrix}
W^{(11)} & \ldots & W^{(1N)} \\
\vdots & \ddots & \vdots \\
W^{(M1)} & \ldots & W^{(MN)}
\end{pmatrix}
\]

\[z(j) = \sum_{i=1}^{M} x^{(i)} W^{(ij)}\]

Want:

\[
\frac{\partial e}{\partial W} = \frac{\partial e}{\partial z} \left[ \begin{array}{c}
\frac{\partial z}{\partial W}
\end{array} \right]_{M \times N, 1 \times N, \ldots, N \times (M \times N)}
\]

\[
\frac{\partial z^{(k)}}{\partial W^{(ij)}} = \mathbb{I}[k = j] x^{(i)}
\]

\[z^{(k)} \text{ depends only on } k\text{th column of } W\]
Matrix-vector multiplication (linear layer)

\( (z^{(1)} \ldots z^{(N)}) = (x^{(1)} \ldots x^{(M)}) \begin{pmatrix} W^{(11)} & \ldots & W^{(1N)} \\ \vdots & \ddots & \vdots \\ W^{(M1)} & \ldots & W^{(MN)} \end{pmatrix} \)

\[ z(j) = \sum_{i=1}^{M} x^{(i)} W^{(ij)} \]

Want:

\[
\frac{\partial e}{\partial W} = \frac{\partial e}{\partial z} \frac{\partial z}{\partial W}
\]

\[
\frac{\partial e}{\partial W^{(ij)}} = \frac{\partial e}{\partial z^{(j)}} x^{(i)}
\]

\[
\frac{\partial e}{\partial W} = \begin{pmatrix}
\frac{\partial e}{\partial z^{(1)}} x^{(1)} & \ldots & \frac{\partial e}{\partial z^{(N)}} x^{(1)} \\
\vdots & \ddots & \vdots \\
\frac{\partial e}{\partial z^{(1)}} x^{(M)} & \ldots & \frac{\partial e}{\partial z^{(N)}} x^{(M)}
\end{pmatrix}
\]
Matrix-vector multiplication (linear layer)

\[
(z^{(1)} \ldots z^{(N)}) = (x^{(1)} \ldots x^{(M)}) \begin{pmatrix} W^{(11)} & \ldots & W^{(1N)} \\ \vdots & \ddots & \vdots \\ W^{(M1)} & \ldots & W^{(MN)} \end{pmatrix} 
\]

\[z(j) = \sum_{i=1}^{M} x^{(i)} W^{(ij)}\]

Want:
\[
\frac{\partial e}{\partial W} = \frac{\partial e}{\partial z} \frac{\partial z}{\partial W}
\]

\[
\frac{\partial e}{\partial W} = x^T \frac{\partial e}{\partial z}
\]
Matrix-vector multiplication (linear layer)

- Summary of backward pass:

\[
f(x, W) = xW
\]

\[
\frac{\partial e}{\partial W} = x^T \frac{\partial e}{\partial z}
\]

\[
\frac{\partial e}{\partial x} = \frac{\partial e}{\partial z} W^T
\]

\[
\frac{\partial e}{\partial z}
\]
General tips

• Derive error signal (upstream gradient) directly, avoid explicit computation of huge local derivatives
• Write out expression for a single element of the Jacobian, then deduce the overall formula
• Keep consistent indexing conventions, order of operations
• Use dimension analysis

• For further reading:
  • Lecture 4 of Stanford 231n and associated links in the syllabus
  • Yes you should understand backprop by Andrej Karpathy