Tutorial on Automatic Differentiation

BIOST 558

Vincent Roulet

UNIVERSITY *of* WASHINGTON

Differentiation Methods

Binary classification
Given sample $(x, y) \in \mathbb{R}^d \times \{-1, 1\}$ want to compute gradient of

$$f : w \to \log(1 + \exp(-yw^\top x))$$

Differentiation Methods

Binary classification
Given sample $(x, y) \in \mathbb{R}^d \times \{-1, 1\}$ want to compute gradient of

$$f : w \to \log(1 + \exp(-yw^\top x))$$

Solutions to compute the gradient:

1. Write down analytic form

$$\nabla f(w) = \frac{-yx}{1 + \exp(-yw^\top x)}$$

Differentiation Methods

Binary classification
Given sample $(x, y) \in \mathbb{R}^d \times \{-1, 1\}$ want to compute gradient of

$$f : w \to \log(1 + \exp(-yw^\top x))$$

Solutions to compute the gradient:

1. Write down analytic form

$$\nabla f(w) = \frac{-yx}{1 + \exp(-yw^\top x)}$$

Pros: Exact formulation, independent of the function evaluation
Cons: Need access to the analytic form of the function

Differentiation Methods

Binary classification
Given sample $(x, y) \in \mathbb{R}^d \times \{-1, 1\}$ wants to compute gradient of

$$f : w \to \log(1 + \exp(-yw^\top x))$$

Solutions to compute the gradient:

1. Write down analytic form
2. Use finite approximation

$$\nabla f(w)^\top d \approx \frac{f(w + \delta d) - f(w)}{\delta} \quad \text{for } 0 < \delta \ll 1$$

Differentiation Methods

Binary classification
Given sample $(x, y) \in \mathbb{R}^d \times \{-1, 1\}$ wants to compute gradient of

$$f : w \to \log(1 + \exp(-yw^\top x))$$

Solutions to compute the gradient:

1. Write down analytic form
2. Use finite approximation

$$\nabla f(w)^\top d \approx \frac{f(w + \delta d) - f(w)}{\delta} \quad \text{for } 0 < \delta \ll 1$$

Pros: Only needs access to the function evaluation of f
Cons: Inexact gradient

Differentiation Methods

Binary classification
Given sample $(x, y) \in \mathbb{R}^d \times \{-1, 1\}$ wants to compute gradient of

$$f : w \to \log(1 + \exp(-yw^\top x))$$

Solutions to compute the gradient:

1. Write down analytic form
2. Use finite approximation
3. Decompose f as successive compositions, use the chain-rule

Differentiation Methods

Binary classification
Given sample $(x, y) \in \mathbb{R}^d \times \{-1, 1\}$ wants to compute gradient of

$$f : w \to \log(1 + \exp(-yw^\top x))$$

Solutions to compute the gradient:

1. Write down analytic form
2. Use finite approximation
3. Decompose f as successive compositions, use the chain-rule

Automatic differentiation

Differentiation Methods

Binary classification
Given sample $(x, y) \in \mathbb{R}^d \times \{-1, 1\}$ wants to compute gradient of

$$f : w \to \log(1 + \exp(-yw^\top x))$$

Solutions to compute the gradient:

1. Write down analytic form
2. Use finite approximation
3. Decompose f as successive compositions, use the chain-rule

Automatic differentiation

Pros: - Only needs access to the function evaluation by compositions
- Exact gradient

Simple Derivative Computation

Consider $\mathbb{R}^d = \mathbb{R}$, a sample $(x, y) = (3.5, 1)$, s.t.

$$f : w_0 \to \log(1 + \exp(-3.5w_0))$$

Simple Derivative Computation

Consider $\mathbb{R}^d = \mathbb{R}$, a sample $(x, y) = (3.5, 1)$, s.t.

$$f : w_0 \to \log(1 + \exp(-3.5 w_0))$$

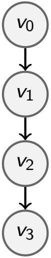
Function decomposition w_0 input, v_k successive evaluations

Simple Derivative Computation

Consider $\mathbb{R}^d = \mathbb{R}$, a sample $(x, y) = (3.5, 1)$, s.t.

$$f : w_0 \to \log(1 + \exp(-3.5 w_0))$$

Function decomposition w_0 input, v_k successive evaluations



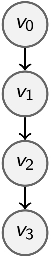
$$\begin{aligned} v_0 &= w_0 \\ v_1 &= -3.5 v_0 \\ v_2 &= 1 + \exp(v_1) \\ v_3 &= \log(v_2) \end{aligned}$$

Simple Derivative Computation

Consider $\mathbb{R}^d = \mathbb{R}$, a sample $(x, y) = (3.5, 1)$, s.t.

$$f : w_0 \to \log(1 + \exp(-3.5 w_0))$$

Function decomposition w_0 input, v_k successive evaluations



$= w_0$

$= -3.5 v_0$

$= 1 + \exp(v_1)$

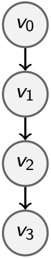
$= \log(v_2) \quad = f(w_0)$

Simple Derivative Computation

Consider $\mathbb{R}^d = \mathbb{R}$, a sample $(x, y) = (3.5, 1)$, s.t.

$$f : w_0 \rightarrow \log(1 + \exp(-3.5 w_0))$$

Function decomposition w_0 input, v_k successive evaluations



$$
\begin{aligned}
v_0 &= w_0 \\
v_1 &= -3.5 v_0 \\
v_2 &= 1 + \exp(v_1) \\
v_3 &= \log(v_2) \quad {\color{red} = f(w_0)}
\end{aligned}
$$

$$f : w_0 \rightarrow g_2 \circ g_1 \circ g_0(w_0)$$

$$
\text{where} \quad
\begin{aligned}
g_0 &: v_0 \rightarrow -3.5 v_0 \\
g_1 &: v_1 \rightarrow 1 + \exp(v_1) \\
g_2 &: v_2 \rightarrow \log(v_2)
\end{aligned}
$$

Chain Rule

Chain rule Given $f(w_0) = g_2 \circ g_1 \circ g_0(w_0)$,

$$f'(w_0) = g_0'(v_0)\, g_1'(v_1)\, g_2'(v_2)$$

where $v_0 = w_0, v_1 = g_0(v_0), v_2 = g_1(v_1)$

Chain Rule

Chain rule Given $f(w_0) = g_2 \circ g_1 \circ g_0(w_0)$,

$$f'(w_0) = g_0'(v_0)\, g_1'(v_1)\, g_2'(v_2)$$

where $v_0 = w_0, v_1 = g_0(v_0), v_2 = g_1(v_1)$

Only need derivatives of elementary functions

Chain Rule

Chain rule Given $f(w_0) = g_2 \circ g_1 \circ g_0(w_0)$,

$$f'(w_0) = g_0'(v_0)\, g_1'(v_1)\, g_2'(v_2)$$

where $v_0 = w_0, v_1 = g_0(v_0), v_2 = g_1(v_1)$

Only need derivatives of elementary functions

Elementary functions

- $v \to av$, $v \to v^k$, $v \to 1/v$
- $v \to \exp(v)$, $v \to \log(v)$, $v \to \cos(v)$, $v \to \sin(v)$
- ...

Forward-Backward Computation

Idea Recursive computations, using $\partial w_0 = \partial v_0$,

$$f'(w_0) = \frac{\partial f}{\partial v_0} = \frac{\partial v_1}{\partial v_0}\,\frac{\partial f}{\partial v_1} = \frac{\partial v_1}{\partial v_0}\,\frac{\partial v_2}{\partial v_1}\,\frac{\partial f}{\partial v_2} = \frac{\partial v_1}{\partial v_0}\,\frac{\partial v_2}{\partial v_1}\,\frac{\partial v_3}{\partial v_2}\,\frac{\partial f}{\partial v_3}$$

Forward-Backward Computation

Idea Recursive computations, using $\partial w_0 = \partial v_0$,

$$f'(w_0) = \frac{\partial f}{\partial v_0} = \frac{\partial v_1}{\partial v_0}\frac{\partial f}{\partial v_1} = \frac{\partial v_1}{\partial v_0}\frac{\partial v_2}{\partial v_1}\frac{\partial f}{\partial v_2} = \frac{\partial v_1}{\partial v_0}\frac{\partial v_2}{\partial v_1}\frac{\partial v_3}{\partial v_2}\frac{\partial f}{\partial v_3}$$

Algorithm

- Compute $\frac{\partial v_{k+1}}{\partial v_k} = g'_k(v_k)$ in a *forward* pass
- Compute $\frac{\partial f}{\partial v_k}$ in a *backward* pass using

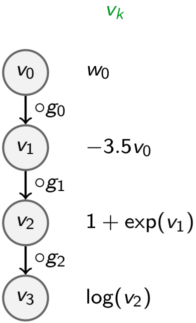
$$\frac{\partial f}{\partial v_k} = \frac{\partial v_{k+1}}{\partial v_k}\frac{\partial f}{\partial v_{k+1}}$$

Simple Derivative Computation

$$f(w_0) = \log(1 + \exp(-3.5w_0)), \qquad v_{k+1} = g_k(v_k) \qquad \lambda_k = \partial f / \partial v_k$$

Simple Derivative Computation

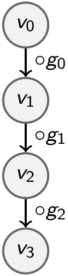
$$f(w_0) = \log(1 + \exp(-3.5 w_0)), \qquad v_{k+1} = g_k(v_k) \qquad \lambda_k = \partial f / \partial v_k$$



	v_k
v_0	w_0
v_1	$-3.5 v_0$
v_2	$1 + \exp(v_1)$
v_3	$\log(v_2)$

Simple Derivative Computation

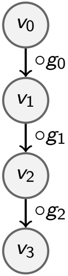
$$f(w_0) = \log(1 + \exp(-3.5 w_0)), \qquad v_{k+1} = g_k(v_k) \qquad \lambda_k = \partial f / \partial v_k$$



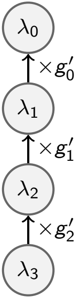
	v_k	$\partial v_k / \partial v_{k-1}$
v_0	w_0	1
v_1	$-3.5 v_0$	-3.5
v_2	$1 + \exp(v_1)$	$\exp(v_1)$
v_3	$\log(v_2)$	$1/v_2$

Simple Derivative Computation

$$f(w_0) = \log(1 + \exp(-3.5 w_0)), \qquad v_{k+1} = g_k(v_k) \qquad \lambda_k = \partial f / \partial v_k$$

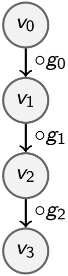


	v_k	$\partial v_k / \partial v_{k-1}$	$\partial f / \partial v_k$
v_0	w_0	1	
v_1	$-3.5 v_0$	-3.5	
v_2	$1 + \exp(v_1)$	$\exp(v_1)$	
v_3	$\log(v_2)$	$1/v_2$	

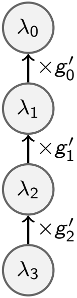


Simple Derivative Computation

$$f(w_0) = \log(1 + \exp(-3.5 w_0)), \qquad v_{k+1} = g_k(v_k) \qquad \lambda_k = \partial f / \partial v_k$$



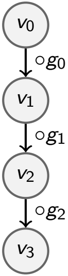
	v_k	$\partial v_k / \partial v_{k-1}$
v_0	w_0	1
v_1	$-3.5 v_0$	-3.5
v_2	$1 + \exp(v_1)$	$\exp(v_1)$
v_3	$\log(v_2)$	$1 / v_2$



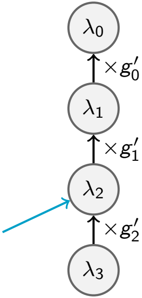
	$\partial f / \partial v_k$
λ_0	
λ_1	
λ_2	
λ_3	1

Simple Derivative Computation

$$f(w_0) = \log(1 + \exp(-3.5w_0)), \qquad v_{k+1} = g_k(v_k) \qquad \lambda_k = \partial f / \partial v_k$$

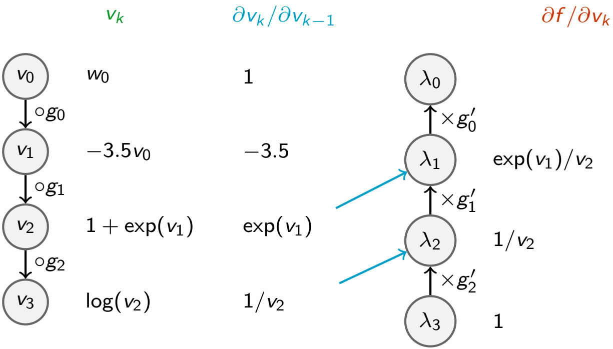


	v_k	$\partial v_k / \partial v_{k-1}$	$\partial f / \partial v_k$
v_0	w_0	1	
v_1	$-3.5v_0$	-3.5	
v_2	$1 + \exp(v_1)$	$\exp(v_1)$	$1/v_2$
v_3	$\log(v_2)$	$1/v_2$	1



Simple Derivative Computation

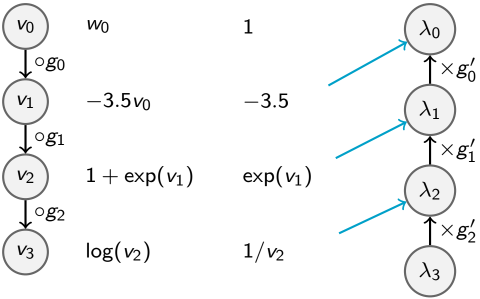
$$f(w_0) = \log(1 + \exp(-3.5 w_0)), \qquad v_{k+1} = g_k(v_k) \qquad \lambda_k = \partial f / \partial v_k$$



	v_k	$\partial v_k / \partial v_{k-1}$		$\partial f / \partial v_k$
v_0	w_0	1	λ_0	
v_1	$-3.5 v_0$	-3.5	λ_1	$\exp(v_1)/v_2$
v_2	$1 + \exp(v_1)$	$\exp(v_1)$	λ_2	$1/v_2$
v_3	$\log(v_2)$	$1/v_2$	λ_3	1

Simple Derivative Computation

$$f(w_0) = \log(1 + \exp(-3.5 w_0)), \qquad v_{k+1} = g_k(v_k) \qquad \lambda_k = \partial f / \partial v_k$$



	v_k	$\partial v_k/\partial v_{k-1}$		$\partial f/\partial v_k$
v_0	w_0	1	λ_0	$-3.5\exp(v_1)/v_2$
v_1	$-3.5 v_0$	-3.5	λ_1	$\exp(v_1)/v_2$
v_2	$1 + \exp(v_1)$	$\exp(v_1)$	λ_2	$1/v_2$
v_3	$\log(v_2)$	$1/v_2$	λ_3	1

Forward-Backward Computation

Forward pass $\frac{\partial v_{k+1}}{\partial v_k}$

- Compute $v_1 = g_0(v_0)$, store $\frac{\partial v_1}{\partial v_0} = g_0'(v_0)$
- Compute $v_2 = g_1(v_1)$, store $\frac{\partial v_2}{\partial v_1} = g_1'(v_1)$,
- Compute $v_3 = g_2(v_2)$, store $\frac{\partial v_3}{\partial v_2} = g_2'(v_2)$

Forward-Backward Computation

Forward pass $\frac{\partial v_{k+1}}{\partial v_k}$

- Compute $v_1 = g_0(v_0)$, store $\frac{\partial v_1}{\partial v_0} = g_0'(v_0)$
- Compute $v_2 = g_1(v_1)$, store $\frac{\partial v_2}{\partial v_1} = g_1'(v_1)$,
- Compute $v_3 = g_2(v_2)$, store $\frac{\partial v_3}{\partial v_2} = g_2'(v_2)$

Backward pass $\frac{\partial f}{\partial v_k}$

- Initialize $\frac{\partial f}{\partial v_3} = 1$
- Compute $\frac{\partial f}{\partial v_2} = \frac{\partial v_3}{\partial v_2} \frac{\partial f}{\partial v_3}$
- Compute $\frac{\partial f}{\partial v_1} = \frac{\partial v_2}{\partial v_1} \frac{\partial f}{\partial v_2}$
- Output $f'(w_0) = \frac{\partial f}{\partial v_0} = \frac{\partial v_1}{\partial v_0} \frac{\partial f}{\partial v_1}$

Gradient Computation

Same forward-backward algorithm, replaces scalar by vectors,

$$f(w_0) = \sum_{i=1}^{n} \log(1 + \exp(-y_i w_0^\top x_i)),\ w_0 \in \mathbb{R}^d,\ x_i \in \mathbb{R}^d,\ y_i \in \{-1, 1\}$$

Gradient Computation

Same forward-backward algorithm, replaces scalar by vectors,

$$f(w_0) = \sum_{i=1}^{n} \log(1 + \exp(-y_i w_0^\top x_i)),\ w_0 \in \mathbb{R}^d,\ x_i \in \mathbb{R}^d,\ y_i \in \{-1, 1\}$$

$$f(w_0) = g_3 \circ g_2 \circ g_1 \circ g_0(w_0)$$

where, denoting $X = (y_1 x_1, \ldots, y_n x_n)^\top$, $\mathbf{1}_n = (1, \ldots, 1)$,

$$v_1 = g_0(v_0) = -Xv_0 \qquad v_3 = g_2(v_2) = \log(v_2)$$

$$v_2 = g_1(v_1) = \mathbf{1}_n + \exp(v_1) \qquad v_4 = g_3(v_3) = \mathbf{1}_n^\top v_3$$

Gradient Computation

Chain rule

$$f(w_0) = g_3 \circ g_2 \circ g_1 \circ g_0(w_0)$$
$$\nabla f(w_0) = \nabla g_0(v_0) \nabla g_1(v_1) \nabla g_2(v_2) \nabla g_3(v_3)$$

where g_2, g_1, g_0 are multivariate functions, e.g., $g_0 : \mathbb{R}^d \rightarrow \mathbb{R}^n$, g_3 is real-valued, i.e,. $g_3 : \mathbb{R}^n \rightarrow \mathbb{R}$

Gradient Computation

Chain rule

$$f(w_0) = g_3 \circ g_2 \circ g_1 \circ g_0(w_0)$$
$$\nabla f(w_0) = \nabla g_0(v_0) \nabla g_1(v_1) \nabla g_2(v_2) \nabla g_3(v_3)$$

where g_2, g_1, g_0 are multivariate functions, e.g., $g_0 : \mathbb{R}^d \to \mathbb{R}^n$, g_3 is real-valued, i.e,. $g_3 : \mathbb{R}^n \to \mathbb{R}$

Consequence: $\nabla g_0(v_0), \nabla g_1(v_1), \nabla g_2(v_2)$ are now matrices,
$\nabla g_3(v_3)$ is a vector

Gradient Computation

Chain rule

$$f(w_0) = g_3 \circ g_2 \circ g_1 \circ g_0(w_0)$$
$$\nabla f(w_0) = \nabla g_0(v_0) \nabla g_1(v_1) \nabla g_2(v_2) \nabla g_3(v_3)$$

where g_2, g_1, g_0 are multivariate functions, e.g., $g_0 : \mathbb{R}^d \to \mathbb{R}^n$, g_3 is real-valued, i.e,. $g_3 : \mathbb{R}^n \to \mathbb{R}$

Consequence: $\nabla g_0(v_0), \nabla g_1(v_1), \nabla g_2(v_2)$ are now matrices,
$\nabla g_3(v_3)$ is a vector

Backward pass $\nabla_{v_k} f$ (vectors)

- Initialize $\nabla_{v2} f = \nabla g_3(v_3)$ (first step amounts to compute a vector)
- For $k = 1, \ldots 0$,
- Compute $\nabla_{v_k} f = \nabla_{v_k} v_{k+1} \nabla_{v_{k+1}} f$
  (iterations are matrix-vector products)
- Output $\nabla f(w_0) = \nabla_{v_0} f$

Gradient for Parametrized Compositions

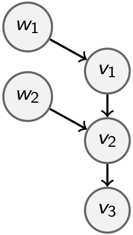
Binary classification with one intermediate parametrized function on $\mathbb{R}$
Given sample $(x, y) = (3.5, 1)$ wants to compute gradient of

$$f : (w_1, w_2) \to (y - w_2 \tanh(xw_1))^2 = (1 - w_2 \tanh(3.5w_1))^2$$

Gradient for Parametrized Compositions

Binary classification with one intermediate parametrized function on $\mathbb{R}$
Given sample $(x, y) = (3.5, 1)$ wants to compute gradient of

$$f : (w_1, w_2) \to (y - w_2 \tanh(xw_1))^2 = (1 - w_2 \tanh(3.5w_1))^2$$



$= \tanh(3.5w_1)$

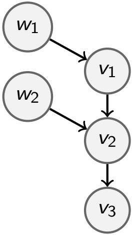
$= w_2 v_1$

$= (1 - v_2)^2$

Gradient for Parametrized Compositions

Binary classification with one intermediate parametrized function on $\mathbb{R}$
Given sample $(x, y) = (3.5, 1)$ wants to compute gradient of

$$f : (w_1, w_2) \to (y - w_2 \tanh(xw_1))^2 = (1 - w_2 \tanh(3.5w_1))^2$$



$$= \tanh(3.5w_1)$$

$$= w_2 v_1$$

$$= (1 - v_2)^2$$

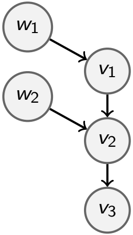
$$\frac{\partial f}{\partial w_1} = \frac{\partial f}{\partial v_1} \frac{\partial v_1}{\partial w_1}$$

$$\frac{\partial f}{\partial w_2} = \frac{\partial f}{\partial v_2} \frac{\partial v_2}{\partial w_2}$$

Gradient for Parametrized Compositions

Binary classification with one intermediate parametrized function on $\mathbb{R}$
Given sample $(x, y) = (3.5, 1)$ wants to compute gradient of

$$f : (w_1, w_2) \to (y - w_2 \tanh(x w_1))^2 = (1 - w_2 \tanh(3.5 w_1))^2$$



$= \tanh(3.5 w_1)$

$= w_2 v_1$

$= (1 - v_2)^2$

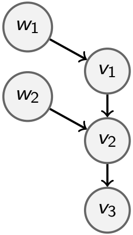
$$\frac{\partial f}{\partial w_1} = \frac{\partial f}{\partial v_1} \frac{\partial v_1}{\partial w_1}$$
$$\frac{\partial f}{\partial w_2} = \frac{\partial f}{\partial v_2} \frac{\partial v_2}{\partial w_2}$$

→ Compute $\frac{\partial f}{\partial v_\ell}$ as previously

Gradient for Parametrized Compositions

Binary classification with one intermediate parametrized function on $\mathbb{R}$
Given sample $(x, y) = (3.5, 1)$ wants to compute gradient of

$$f : (w_1, w_2) \to (y - w_2 \tanh(x w_1))^2 = (1 - w_2 \tanh(3.5 w_1))^2$$



$= \tanh(3.5 w_1)$

$= w_2 v_1$

$= (1 - v_2)^2$

$$\frac{\partial f}{\partial w_1} = \frac{\partial f}{\partial v_1} \frac{\partial v_1}{\partial w_1}$$
$$\frac{\partial f}{\partial w_2} = \frac{\partial f}{\partial v_2} \frac{\partial v_2}{\partial w_2}$$

→ Compute $\frac{\partial f}{\partial v_\ell}$ as previously
→ At node v_ℓ, output

$$\frac{\partial f}{\partial w_\ell} = \frac{\partial f}{\partial v_\ell} \frac{\partial v_\ell}{\partial w_\ell}$$

Forward-Backward Computation

Forward pass $\frac{\partial v_{k+1}}{\partial v_k}, \frac{\partial v_k}{\partial w_k}$

- Compute $v_1 = g_0(w_1)$, store $\frac{\partial v_1}{\partial w_1}$
- Compute $v_2 = g_1(v_1, w_2)$, store $\frac{\partial v_2}{\partial w_2}$, $\frac{\partial v_2}{\partial v_1}$,
- Compute $v_3 = g_2(v_2)$, store $\frac{\partial v_3}{\partial v_2}$

Forward-Backward Computation

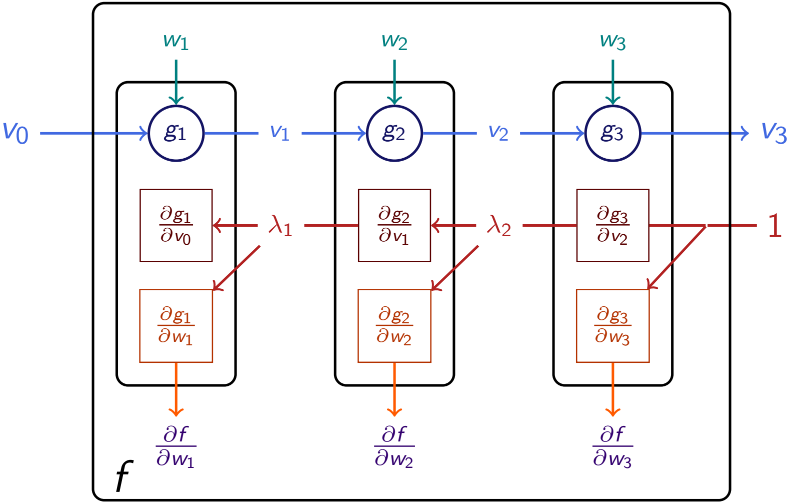
Forward pass $\frac{\partial v_{k+1}}{\partial v_k}, \frac{\partial v_k}{\partial w_k}$

- Compute $v_1 = g_0(w_1)$, store $\frac{\partial v_1}{\partial w_1}$
- Compute $v_2 = g_1(v_1, w_2)$, store $\frac{\partial v_2}{\partial w_2}$, $\frac{\partial v_2}{\partial v_1}$,
- Compute $v_3 = g_2(v_2)$, store $\frac{\partial v_3}{\partial v_2}$

Backward pass $\frac{\partial f}{\partial v_k}, \frac{\partial f}{\partial w_k}$

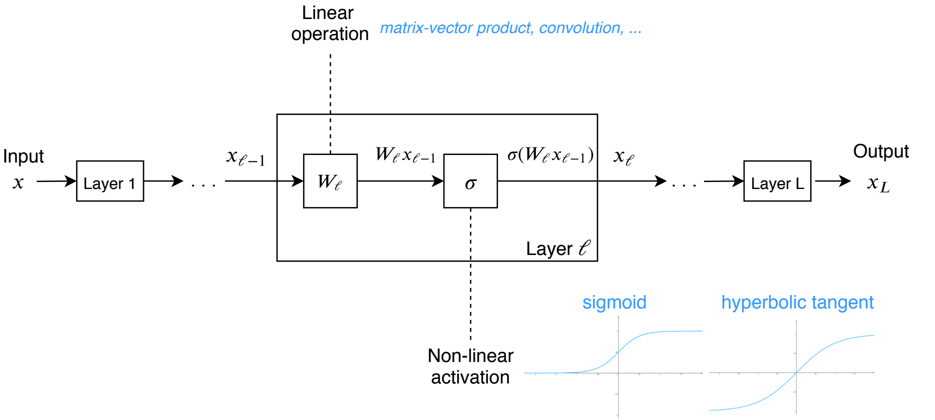
- Initialize $\frac{\partial f}{\partial v_3} = 1$
- For $k = 2, \ldots 0$,
-     Compute $\frac{\partial f}{\partial v_k} = \frac{\partial v_{k+1}}{\partial v_k} \frac{\partial f}{\partial v_{k+1}}$
-     Output $\frac{\partial f}{\partial w_k} = \frac{\partial f}{\partial v_k} \frac{\partial v_k}{\partial w_k}$

Automatic differentiation scheme


w_1
w_2
w_3
v_0
g_1
v_1
g_2
v_2
g_3
v_3
$\frac{\partial g_1}{\partial v_0}$
λ_1
$\frac{\partial g_2}{\partial v_1}$
λ_2
$\frac{\partial g_3}{\partial v_2}$
1
$\frac{\partial g_1}{\partial w_1}$
$\frac{\partial g_2}{\partial w_2}$
$\frac{\partial g_3}{\partial w_3}$
$\frac{\partial f}{\partial w_1}$
$\frac{\partial f}{\partial w_2}$
$\frac{\partial f}{\partial w_3}$
f

Automatic differentiation for $f(v_0, w_1, w_2, w_3) = v_3$

Deep Neural Network



Linear operation

matrix-vector product, convolution, ...

Input x

Layer 1

$\dots$

$x_{\ell-1}$

W_ℓ

$W_\ell x_{\ell-1}$

σ

$\sigma(W_\ell x_{\ell-1})$

x_ℓ

Layer ℓ

$\dots$

Layer L

Output x_L

Non-linear activation

sigmoid

hyperbolic tangent

$\sigma(x)=(1+\exp(-x))^{-1}$ $\sigma(x)=(\exp(2x)-1)/(\exp(2x)+1)$

Deep Neural Network

Deep neural network structure
A deep neural network transforms an input $x = x_0$ using

$$x_\ell = \sigma_\ell(W_\ell \cdot x_{\ell-1}) \tag{Layer ℓ}$$

where σ_ℓ is the activation function, W_ℓ are the weights of the layer

Deep Neural Network

Deep neural network structure
A deep neural network transforms an input $x = x_0$ using

$$x_\ell = \sigma_\ell(W_\ell \cdot x_{\ell-1}) \qquad \text{(Layer } \ell\text{)}$$

where σ_ℓ is the activation function, W_ℓ are the weights of the layer

Objective

$$\min_{W=(W_0,\dots,W_L)} \quad \frac{1}{n}\sum_{i=1}^{n} f^{(i)}(W) = \frac{1}{n}\sum_{i=1}^{n} f\left(y^{(i)}, x_L^{(i)}(W_0, \dots, W_L)\right)$$

Deep Neural Network

Deep neural network structure
A deep neural network transforms an input $x = x_0$ using

$$x_\ell = \sigma_\ell(W_\ell \cdot x_{\ell-1}) \qquad \text{(Layer } \ell\text{)}$$

where σ_ℓ is the activation function, W_ℓ are the weights of the layer

Objective

$$\min_{W=(W_0,\ldots,W_L)} \quad \frac{1}{n}\sum_{i=1}^{n} f^{(i)}(W) = \frac{1}{n}\sum_{i=1}^{n} f\left(y^{(i)}, x_L^{(i)}(W_0,\ldots,W_L)\right)$$

with stochastic gradient descent

$$W \leftarrow W - \gamma \nabla f^{(i)}(W)$$