Chapter ML:IV (continued)

IV. Neural Networks

- Perceptron Learning
- Multilayer Perceptron
- Advanced MLPs
- Automatic Gradient Computation

Advanced MLPs

Softmax Output

Advanced MLPs

Loss Functions

Advanced MLPs

Activation Functions

Advanced MLPs

Regularization

Advanced MLPs

Momentum Term

Momentum idea: a weight adaptation in iteration t considers the adaptation in iteration $t-1$:

$$\Delta W^{\mathsf{o}}(t) = \eta \cdot (\boldsymbol{\delta}^{\mathsf{o}} \otimes \mathbf{y}^{\mathsf{h}}(\mathbf{x})|_{1,\ldots,l}) + \alpha \cdot \Delta W^{\mathsf{o}}(t-1)$$

$$\Delta W^{\mathsf{h}}(t) = \eta \cdot (\boldsymbol{\delta}^{\mathsf{h}} \otimes \mathbf{x}) + \alpha \cdot \Delta W^{\mathsf{h}}(t-1)$$

$$\Delta W^{\mathsf{h}_s}(t) = \eta \cdot (\boldsymbol{\delta}^{\mathsf{h}_s} \otimes \mathbf{y}^{\mathsf{h}_{s-1}}(\mathbf{x})|_{1,\ldots,l_{s-1}}) + \alpha \cdot \Delta W^{\mathsf{h}_s}(t-1), \quad s = d, d-1, \ldots, 2$$

$$\Delta W^{\mathsf{h}_1}(t) = \eta \cdot (\boldsymbol{\delta}^{\mathsf{h}_1} \otimes \mathbf{x}) + \alpha \cdot \Delta W^{\mathsf{h}_1}(t-1)$$

The term α, $0 \leq \alpha < 1$, is called "momentum".

Effects:

- Due the "adaptation inertia" local minima can be overcome.
- If the direction of the descent does not change, the adaptation increment and, as a consequence, the speed of convergence is increased.

Remarks:

- Recap. The symbol »$\otimes$« denotes the dyadic product, also called outer product or tensor product. The dyadic product takes two vectors and returns a second order tensor, called a dyadic in this context: $\mathbf{v} \otimes \mathbf{w} \equiv \mathbf{v}\mathbf{w}^T$. [Wikipedia]

Automatic Gradient Computation

The IGD Algorithm for MLP with arbitrary model and objective functions

Algorithm:	$\text{IGD}_{\text{MLP}_d}$	Incremental Gradient Descent for the (modified) d-layer MLP
Input:	D	Training examples $(\mathbf{x}, \mathbf{c}(\mathbf{x}))$ with $\mathbf{x} \in \mathbf{R}^p$, $\mathbf{c}(\mathbf{x}) \in \{0,1\}^k$.
	η	learning rate, a small positive constant.
	$\mathcal{L}$, ℓ, R	optimization objective.
Output:	$W^{\text{h}_1}, \ldots, W^{\text{h}_d}$	Weight matrices of the d layers. (= hypothesis)

1. FOR $s = 1$ TO d DO *initialize_random_weights*(W^{h_s}) ENDDO, $t = 0$
2. **REPEAT**
3. $\quad t = t + 1$
4. $\quad$ FOREACH $(\mathbf{x}, \mathbf{c}(\mathbf{x})) \in D$ DO
5. $\quad\quad \mathbf{y}^{\text{h}_1}(\mathbf{x}) = \left(\begin{smallmatrix}1\\ \mathbf{tanh}(W^{\text{h}_1}\mathbf{x})\end{smallmatrix}\right)$ // forward propagation; $\mathbf{x}$ is extended by $x_0 = 1$
$\quad\quad$ FOR $s = 2$ TO $d{-}1$ DO $\mathbf{y}^{\text{h}_s}(\mathbf{x}) = \left(\begin{smallmatrix}1\\ \text{ReLU}(W^{\text{h}_s}\mathbf{y}^{\text{h}_{s-1}}(\mathbf{x}))\end{smallmatrix}\right)$ ENDDO
$\quad\quad \mathbf{y}(\mathbf{x}) = \text{softmax}(W^{\text{h}_d}\mathbf{y}^{\text{h}_{d-1}}(\mathbf{x}))$
6. $\quad\quad \mathcal{L}(\mathbf{w}) = \ell(c(\mathbf{x}), y(\mathbf{x})) + \text{R}(\mathbf{w})$ // loss computation
$\quad\quad$??? // backpropagation
7. $\quad\quad$ FOR $s = 1$ TO d DO $\Delta W^{\text{h}_s} = \eta \cdot (???)$ ENDDO // weight update
8. $\quad\quad$ FOR $s = 1$ TO d DO $W^{\text{h}_s} = W^{\text{h}_s} + \Delta W^{\text{h}_s}$ ENDDO
9. $\quad$ ENDDO
10. **UNTIL**(*convergence*$(\mathbf{c}(D), \mathbf{y}(D))$ **OR** $t > t_{\text{max}}$)
11. *return*$(W^{\text{h}_1}, \ldots, W^{\text{h}_d})$

Automatic Gradient Computation

The IGD Algorithm for MLP with arbitrary model and objective functions

Algorithm:	$\text{IGD}_{\text{MLP}_d}$	Incremental Gradient Descent for the (modified) d-layer MLP
Input:	D	Training examples $(\mathbf{x}, \mathbf{c}(\mathbf{x}))$ with $\mathbf{x} \in \mathbf{R}^p$, $\mathbf{c}(\mathbf{x}) \in \{0,1\}^k$.
	η	learning rate, a small positive constant.
	$\mathcal{L}$, ℓ, R	optimization objective.
Output:	$W^{\text{h}_1}, \ldots, W^{\text{h}_d}$	Weight matrices of the d layers. (= hypothesis)

```
 1.  FOR s = 1 TO d DO initialize_random_weights(W^{h_s}) ENDDO, t = 0
 2.  REPEAT
 3.    t = t + 1
 4.    FOREACH (x, c(x)) ∈ D DO
 5.      y^{h_1}(x) = (1 ; tanh(W^{h_1} x))   // forward propagation; x is extended by x_0 = 1
         FOR s = 2 TO d−1 DO y^{h_s}(x) = (1 ; ReLU(W^{h_s} y^{h_{s−1}}(x))) ENDDO
         y(x) = softmax(W^{h_d} y^{h_{d−1}}(x))
 6.      L(w) = ℓ(c(x), y(x)) + R(w)   // loss computation
         ∇L(w) = autodiff(L, w)   // backpropagation
 7.      FOR s = 1 TO d DO ΔW^{h_s} = η · (∇L^{h_s}(w)) ENDDO // weight update
 8.      FOR s = 1 TO d DO W^{h_s} = W^{h_s} + ΔW^{h_s} ENDDO
 9.    ENDDO
10.  UNTIL(convergence(c(D), y(D)) OR t > t_max)
11.  return(W^{h_1}, ..., W^{h_d})
```

Formulas of steps 5–7:

$$\mathbf{y}^{\text{h}_1}(\mathbf{x}) = \begin{pmatrix} 1 \\ \tanh(W^{\text{h}_1}\mathbf{x}) \end{pmatrix}, \quad \mathbf{y}^{\text{h}_s}(\mathbf{x}) = \begin{pmatrix} 1 \\ \text{ReLU}(W^{\text{h}_s}\mathbf{y}^{\text{h}_{s-1}}(\mathbf{x})) \end{pmatrix}, \quad \mathbf{y}(\mathbf{x}) = \text{softmax}(W^{\text{h}_d}\mathbf{y}^{\text{h}_{d-1}}(\mathbf{x}))$$

$$\mathcal{L}(\mathbf{w}) = \ell(c(\mathbf{x}), y(\mathbf{x})) + \text{R}(\mathbf{w}), \quad \nabla\mathcal{L}(\mathbf{w}) = \text{autodiff}(\mathcal{L}, \mathbf{w}), \quad \Delta W^{\text{h}_s} = \eta \cdot (\nabla\mathcal{L}^{\text{h}_s}(\mathbf{w}))$$

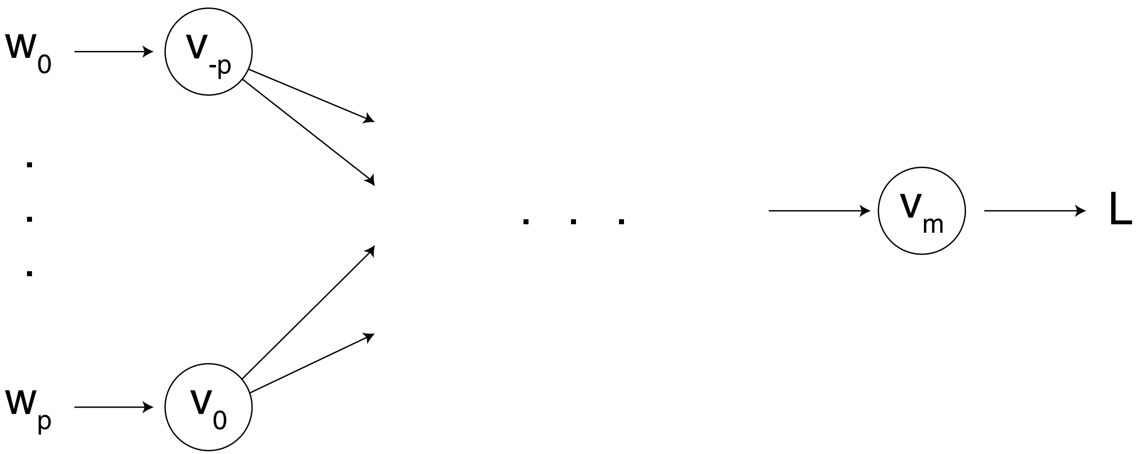
Automatic Gradient Computation

Reverse-Mode Automatic Differentiation in Arbitrary Computational Graphs

Reverse-mode AD corresponds to a generalized backpropagation algorithm.

Let $\mathcal{L}(w_1, \ldots, w_p)$ be the function to be differentiated.

- Consider $\mathcal{L}$ as a computational graph of elementary operations, assigning each intermediate result to a variable v_i with $-p \leq i \leq m$
  (naming convention: $v_{-p\ldots0}$ for inputs, $v_{1\ldots m-1}$ for intermediate variables, $v_m \equiv \mathcal{L}$ for the output)

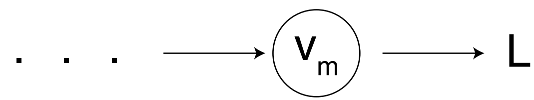


Automatic Gradient Computation

Patterns in computation graphs

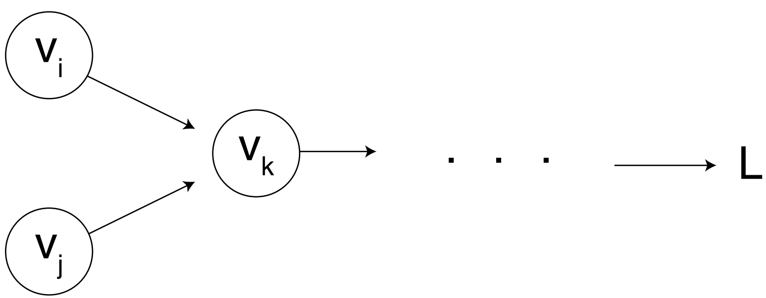
- For each intermediate variable v_i, an adjoint value $\bar{v}_i = \frac{\partial y}{\partial v_i}$ is computed based on its descendants in the computation graph

(1)



$$\bar{v}_m = \frac{\partial L}{\partial v_m} = \frac{\partial v_m}{\partial v_m} = 1$$

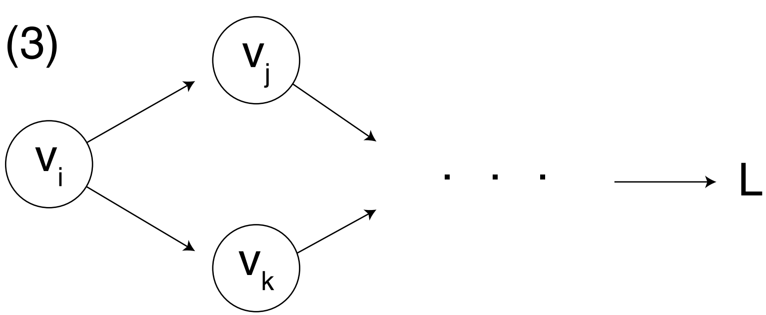
(2)



$$\bar{v}_i = \frac{\partial L}{\partial v_i} = \frac{\partial L}{\partial v_k} \cdot \frac{\partial v_k}{\partial v_i} = \bar{v}_k \cdot \frac{\partial v_k}{\partial v_i}$$

$$\bar{v}_j = \frac{\partial L}{\partial v_j} = \frac{\partial L}{\partial v_k} \cdot \frac{\partial v_k}{\partial v_j} = \bar{v}_k \cdot \frac{\partial v_k}{\partial v_j}$$

(3)



$$\bar{v}_i = \bar{v}_j \frac{\partial v_j}{\partial v_i} + \bar{v}_k \frac{\partial v_k}{\partial v_i}$$

Remarks

- Adjoints are computed in reverse, starting from $\bar{v}_m$
- For any step $v_j = g(\ldots, v_i, \ldots)$ in the graph, the local gradients $\frac{\partial g}{\partial v_i}$ must be computable

Automatic Gradient Computation

Example: Setting

Consider the RSS loss for a simple logistic regression model and a very small dataset.

Dataset: $D = \{((1, 1.5)^T, 0), ((1.5, -1)^T, 1)\}$

Model function: $y(x) = \sigma(\mathbf{w}^T\mathbf{x})$

Loss function: $\mathcal{L}(\mathbf{w}) = L_2(\mathbf{w}) = \sum_{(\mathbf{x}, c(\mathbf{x})) \in D} (c(\mathbf{x}) - y(\mathbf{x}))^2$

$\mathcal{L}(\mathbf{w})$ is the objective function to be minimized, and hence what we want to compute the derivative of; everything except $\mathbf{w}$ is held constant.

Given the setting above, we can rewrite $\mathcal{L}$ as:

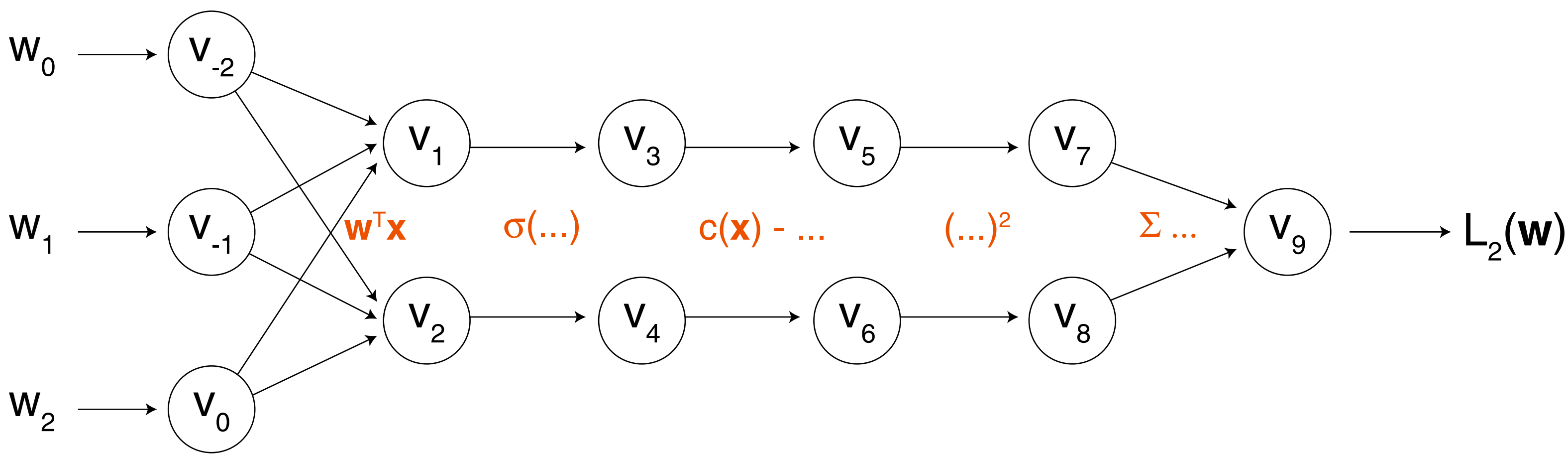
$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= (c(\mathbf{x}_1) - \sigma(\mathbf{w}^T\mathbf{x}_1))^2 + (c(\mathbf{x}_2) - \sigma(\mathbf{w}^T\mathbf{x}_2))^2 \\ &= (-\sigma(w_0 + w_1 + 1.5w_2))^2 + (1 - \sigma(w_0 + 1.5w_1 - w_2))^2\end{aligned}$$

Using reverse-mode automatic differentiation, we'll simultaneously evaluate the loss and its derivative at $\mathbf{w} = (-1, 1.5, 0.5)^T$.

Automatic Gradient Computation

Example: Computation Graph

$$\mathcal{L}(\mathbf{w}) = \underbrace{\overbrace{(\underbrace{-\overbrace{\sigma(\underbrace{w_0+w_1+1.5w_2}_{v_1})}^{v_3}}_{v_5})^2}^{v_7} + \overbrace{(\underbrace{1-\overbrace{\sigma(\underbrace{w_0+1.5w_1-w_2}_{v_2})}^{v_4}}_{v_6})^2}^{v_8}}_{v_9}$$



Automatic Gradient Computation

Example: Forward and Reverse Trace

$$\mathcal{L}(\mathbf{w}) = \underbrace{\overbrace{(\underbrace{-\overbrace{\sigma(\underbrace{w_0+w_1+1.5w_2}_{v_1})}^{v_3}}_{v_5})^2}^{v_7} + \overbrace{(\underbrace{1-\overbrace{\sigma(\underbrace{w_0+1.5w_1-w_2}_{v_2})}^{v_4}}_{v_6})^2}^{v_8}}_{v_9} \qquad \text{at} \qquad \mathbf{w} = (-1, 1.5, 0.5)^T$$

Forward Primal Trace		Reverse Adjoint Trace
$v_{-2} = w_0$	$= -1$	
$v_{-1} = w_1$	$= 1.5$	
$v_0 = w_2$	$= 0.5$	
$v_1 = v_{-2} + v_{-1} + 1.5 \cdot v_0$	$= 1.25$	
$v_2 = v_{-2} + 1.5 \cdot v_{-1} - v_0$	$= 0.75$	
$v_3 = \sigma(v_1)$	$= 0.78$	
$v_4 = \sigma(v_2)$	$= 0.68$	
$v_5 = 0 - v_3$	$= -0.78$	
$v_6 = 1 - v_4$	$= 0.32$	
$v_7 = v_5^2$	$= 0.61$	
$v_8 = v_6^2$	$= 0.1$	
$v_9 = v_7 + v_8$	$= 0.71$	
$L_2 = v_9$	$= 0.71$	

Automatic Gradient Computation

Example: Forward and Reverse Trace

$$\mathcal{L}(\mathbf{w}) = \underbrace{\overbrace{(\underbrace{- \overbrace{\sigma(\underbrace{w_0 + w_1 + 1.5w_2}_{v_1})}^{v_3})}_{v_5}{}^2}^{v_7} + \overbrace{(\underbrace{1 - \overbrace{\sigma(\underbrace{w_0 + 1.5w_1 - w_2}_{v_2})}^{v_4})}_{v_6}{}^2}^{v_8}}_{v_9} \qquad \text{at} \qquad \mathbf{w} = (-1, 1.5, 0.5)^T$$

Forward Primal Trace		Reverse Adjoint Trace	
$v_{-2} = w_0$	$= -1$		
$v_{-1} = w_1$	$= 1.5$		
$v_0 = w_2$	$= 0.5$		
$v_1 = v_{-2} + v_{-1} + 1.5 \cdot v_0$	$= 1.25$		
$v_2 = v_{-2} + 1.5 \cdot v_{-1} - v_0$	$= 0.75$		
$v_3 = \sigma(v_1)$	$= 0.78$		
$v_4 = \sigma(v_2)$	$= 0.68$		
$v_5 = 0 - v_3$	$= -0.78$		
$v_6 = 1 - v_4$	$= 0.32$		
$v_7 = v_5^2$	$= 0.61$		
$v_8 = v_6^2$	$= 0.1$		
$v_9 = v_7 + v_8$	$= 0.71$		
$L_2 = v_9$	$= 0.71$	$\bar{v}_9 = \frac{\partial L_2}{\partial v_9}$	$= 1$

Automatic Gradient Computation

Example: Forward and Reverse Trace

$$\mathcal{L}(\mathbf{w}) = \underbrace{\overbrace{(\underbrace{-\overbrace{\sigma(\underbrace{w_0 + w_1 + 1.5w_2}_{v_1})}^{v_3}}_{v_5})^2}^{v_7} + \overbrace{(\underbrace{1 - \overbrace{\sigma(\underbrace{w_0 + 1.5w_1 - w_2}_{v_2})}^{v_4}}_{v_6})^2}^{v_8}}_{v_9} \quad \text{at} \quad \mathbf{w} = (-1, 1.5, 0.5)^T$$

Forward Primal Trace		Reverse Adjoint Trace	
$v_{-2} = w_0$	$= -1$		
$v_{-1} = w_1$	$= 1.5$		
$v_0 = w_2$	$= 0.5$		
$v_1 = v_{-2} + v_{-1} + 1.5 \cdot v_0$	$= 1.25$		
$v_2 = v_{-2} + 1.5 \cdot v_{-1} - v_0$	$= 0.75$		
$v_3 = \sigma(v_1)$	$= 0.78$		
$v_4 = \sigma(v_2)$	$= 0.68$		
$v_5 = 0 - v_3$	$= -0.78$		
$v_6 = 1 - v_4$	$= 0.32$		
$v_7 = v_5^2$	$= 0.61$		
$v_8 = v_6^2$	$= 0.1$		
$v_9 = v_7 + v_8$	$= 0.71$	$\bar{v}_8 = \bar{v}_9 \cdot \frac{\partial v_9}{\partial v_8} = 1 \cdot 1$	$= 1$
		$\bar{v}_7 = \bar{v}_9 \cdot \frac{\partial v_9}{\partial v_7} = 1 \cdot 1$	$= 1$
$L_2 = v_9$	$= 0.71$	$\bar{v}_9 = \frac{\partial L_2}{\partial v_9}$	$= 1$

Automatic Gradient Computation

Example: Forward and Reverse Trace

$$\mathcal{L}(\mathbf{w}) = \underbrace{\overbrace{(-\overbrace{\sigma(\underbrace{w_0 + w_1 + 1.5w_2}_{v_1})}^{v_3})^2}^{v_7}}_{v_5} + \overbrace{(1 - \overbrace{\sigma(\underbrace{w_0 + 1.5w_1 - w_2}_{v_2})}^{v_4})^2}^{v_8}_{v_6} \quad \text{at} \quad \mathbf{w} = (-1, 1.5, 0.5)^T$$

(the whole expression is v_9)

Forward Primal Trace		Reverse Adjoint Trace	
$v_{-2} = w_0$	$= -1$		
$v_{-1} = w_1$	$= 1.5$		
$v_0 = w_2$	$= 0.5$		
$v_1 = v_{-2} + v_{-1} + 1.5 \cdot v_0$	$= 1.25$		
$v_2 = v_{-2} + 1.5 \cdot v_{-1} - v_0$	$= 0.75$		
$v_3 = \sigma(v_1)$	$= 0.78$		
$v_4 = \sigma(v_2)$	$= 0.68$		
$v_5 = 0 - v_3$	$= -0.78$	$\bar{v}_3 = \bar{v}_5 \cdot (-1)$	$= 1.55$
$v_6 = 1 - v_4$	$= 0.32$	$\bar{v}_4 = \bar{v}_6 \cdot (-1)$	$= -0.64$
$v_7 = v_5^2$	$= 0.61$	$\bar{v}_5 = \bar{v}_7 \cdot \frac{\partial v_7}{\partial v_5} = \bar{v}_7 \cdot 2v_5$	$= -1.55$
$v_8 = v_6^2$	$= 0.1$	$\bar{v}_6 = \bar{v}_8 \cdot \frac{\partial v_8}{\partial v_6} = \bar{v}_8 \cdot 2v_6$	$= 0.64$
$v_9 = v_7 + v_8$	$= 0.71$	$\bar{v}_8 = \bar{v}_9 \cdot \frac{\partial v_9}{\partial v_8} = 1 \cdot 1$	$= 1$
		$\bar{v}_7 = \bar{v}_9 \cdot \frac{\partial v_9}{\partial v_7} = 1 \cdot 1$	$= 1$
$L_2 = v_9$	$= 0.71$	$\bar{v}_9 = \frac{\partial L_2}{\partial v_9}$	$= 1$

Automatic Gradient Computation

Example: Forward and Reverse Trace

$$\mathcal{L}(\mathbf{w}) = \underbrace{\overbrace{(-\overbrace{\sigma(\underbrace{w_0 + w_1 + 1.5w_2}_{v_1})}^{v_3})^2}^{v_7}}_{} + \overbrace{(1 - \overbrace{\sigma(\underbrace{w_0 + 1.5w_1 - w_2}_{v_2})}^{v_4})^2}^{v_8}$$

with $v_5 = (-v_3)$, $v_6 = (1 - v_4)$, and v_9 the whole expression, at $\mathbf{w} = (-1, 1.5, 0.5)^T$

Forward Primal Trace		Reverse Adjoint Trace	
$v_{-2} = w_0$	$= -1$		
$v_{-1} = w_1$	$= 1.5$		
$v_0 = w_2$	$= 0.5$		
$v_1 = v_{-2} + v_{-1} + 1.5 \cdot v_0$	$= 1.25$		
$v_2 = v_{-2} + 1.5 \cdot v_{-1} - v_0$	$= 0.75$		
$v_3 = \sigma(v_1)$	$= 0.78$	$\bar{v}_1 = \bar{v}_3 \cdot \sigma(v_1) \cdot (1 - \sigma(v_1))$	$= 0.27$
$v_4 = \sigma(v_2)$	$= 0.68$	$\bar{v}_2 = \bar{v}_4 \cdot \sigma(v_2) \cdot (1 - \sigma(v_2))$	$= -0.14$
$v_5 = 0 - v_3$	$= -0.78$	$\bar{v}_3 = \bar{v}_5 \cdot (-1)$	$= 1.55$
$v_6 = 1 - v_4$	$= 0.32$	$\bar{v}_4 = \bar{v}_6 \cdot (-1)$	$= -0.64$
$v_7 = v_5^2$	$= 0.61$	$\bar{v}_5 = \bar{v}_7 \cdot \frac{\partial v_7}{\partial v_5} = \bar{v}_7 \cdot 2v_5$	$= -1.55$
$v_8 = v_6^2$	$= 0.1$	$\bar{v}_6 = \bar{v}_8 \cdot \frac{\partial v_8}{\partial v_6} = \bar{v}_8 \cdot 2v_6$	$= 0.64$
$v_9 = v_7 + v_8$	$= 0.71$	$\bar{v}_8 = \bar{v}_9 \cdot \frac{\partial v_9}{\partial v_8} = 1 \cdot 1$	$= 1$
		$\bar{v}_7 = \bar{v}_9 \cdot \frac{\partial v_9}{\partial v_7} = 1 \cdot 1$	$= 1$
$L_2 = v_9$	$= 0.71$	$\bar{v}_9 = \frac{\partial L_2}{\partial v_9}$	$= 1$

Automatic Gradient Computation

Example: Forward and Reverse Trace

$$\mathcal{L}(\mathbf{w}) = \underbrace{\overbrace{(\underbrace{-\overbrace{\sigma(\underbrace{w_0 + w_1 + 1.5w_2}_{v_1})}^{v_3}}_{v_5})^2}^{v_7} + \overbrace{(\underbrace{1 - \overbrace{\sigma(\underbrace{w_0 + 1.5w_1 - w_2}_{v_2})}^{v_4}}_{v_6})^2}^{v_8}}_{v_9}$$

at $\mathbf{w} = (-1, 1.5, 0.5)^T$

Forward Primal Trace		Reverse Adjoint Trace	
$v_{-2} = w_0$	$= -1$		
$v_{-1} = w_1$	$= 1.5$		
$v_0 = w_2$	$= 0.5$		
$v_1 = v_{-2} + v_{-1} + 1.5 \cdot v_0$	$= 1.25$	$\bar{v}_0 = \bar{v}_0 + \bar{v}_1 \cdot 1.5$	$= 0.54$
		$\bar{v}_{-1} = \bar{v}_{-1} + \bar{v}_1$	$= 0.06$
		$\bar{v}_{-2} = \bar{v}_{-2} + \bar{v}_1$	$= 0.13$
$v_2 = v_{-2} + 1.5 \cdot v_{-1} - v_0$	$= 0.75$	$\bar{v}_0 = \bar{v}_2 \cdot (-1)$	$= 0.14$
		$\bar{v}_{-1} = \bar{v}_2 \cdot 1.5$	$= -0.28$
		$\bar{v}_{-2} = \bar{v}_2$	$= -0.14$
$v_3 = \sigma(v_1)$	$= 0.78$	$\bar{v}_1 = \bar{v}_3 \cdot \sigma(v_1) \cdot (1 - \sigma(v_1))$	$= 0.27$
$v_4 = \sigma(v_2)$	$= 0.68$	$\bar{v}_2 = \bar{v}_4 \cdot \sigma(v_2) \cdot (1 - \sigma(v_2))$	$= -0.14$
$v_5 = 0 - v_3$	$= -0.78$	$\bar{v}_3 = \bar{v}_5 \cdot (-1)$	$= 1.55$
$v_6 = 1 - v_4$	$= 0.32$	$\bar{v}_4 = \bar{v}_6 \cdot (-1)$	$= -0.64$
$v_7 = v_5^2$	$= 0.61$	$\bar{v}_5 = \bar{v}_7 \cdot \frac{\partial v_7}{\partial v_5} = \bar{v}_7 \cdot 2v_5$	$= -1.55$
$v_8 = v_6^2$	$= 0.1$	$\bar{v}_6 = \bar{v}_8 \cdot \frac{\partial v_8}{\partial v_6} = \bar{v}_8 \cdot 2v_6$	$= 0.64$
$v_9 = v_7 + v_8$	$= 0.71$	$\bar{v}_8 = \bar{v}_9 \cdot \frac{\partial v_9}{\partial v_8} = 1 \cdot 1$	$= 1$
		$\bar{v}_7 = \bar{v}_9 \cdot \frac{\partial v_9}{\partial v_7} = 1 \cdot 1$	$= 1$
$L_2 = v_9$	$= 0.71$	$\bar{v}_9 = \frac{\partial L_2}{\partial v_9}$	$= 1$

Automatic Gradient Computation

Example: Forward and Reverse Trace

$$\mathcal{L}(\mathbf{w}) = \underbrace{\overbrace{(\underbrace{-\overbrace{\sigma(\underbrace{w_0+w_1+1.5w_2}_{v_1})}^{v_3}}_{v_5})^2}^{v_7} + \overbrace{(\underbrace{1-\overbrace{\sigma(\underbrace{w_0+1.5w_1-w_2}_{v_2})}^{v_4}}_{v_6})^2}^{v_8}}_{v_9} \qquad \text{at} \qquad \mathbf{w} = (-1, 1.5, 0.5)^T$$

Forward Primal Trace		Reverse Adjoint Trace	
$v_{-2} = w_0$	$= -1$	$\bar{w}_0 = \frac{\partial L_2}{\partial w_0} = \bar{v}_{-2}$	$= \mathbf{0.13}$
$v_{-1} = w_1$	$= 1.5$	$\bar{w}_1 = \frac{\partial L_2}{\partial w_1} = \bar{v}_{-1}$	$= \mathbf{0.06}$
$v_0 = w_2$	$= 0.5$	$\bar{w}_2 = \frac{\partial L_2}{\partial w_2} = \bar{v}_0$	$= \mathbf{0.54}$
$v_1 = v_{-2} + v_{-1} + 1.5 \cdot v_0$	$= 1.25$	$\bar{v}_0 = \bar{v}_0 + \bar{v}_1 \cdot 1.5$	$= 0.54$
		$\bar{v}_{-1} = \bar{v}_{-1} + \bar{v}_1$	$= 0.06$
		$\bar{v}_{-2} = \bar{v}_{-2} + \bar{v}_1$	$= 0.13$
$v_2 = v_{-2} + 1.5 \cdot v_{-1} - v_0$	$= 0.75$	$\bar{v}_0 = \bar{v}_2 \cdot (-1)$	$= 0.14$
		$\bar{v}_{-1} = \bar{v}_2 \cdot 1.5$	$= -0.28$
		$\bar{v}_{-2} = \bar{v}_2$	$= -0.14$
$v_3 = \sigma(v_1)$	$= 0.78$	$\bar{v}_1 = \bar{v}_3 \cdot \sigma(v_1) \cdot (1 - \sigma(v_1))$	$= 0.27$
$v_4 = \sigma(v_2)$	$= 0.68$	$\bar{v}_2 = \bar{v}_4 \cdot \sigma(v_2) \cdot (1 - \sigma(v_2))$	$= -0.14$
$v_5 = 0 - v_3$	$= -0.78$	$\bar{v}_3 = \bar{v}_5 \cdot (-1)$	$= 1.55$
$v_6 = 1 - v_4$	$= 0.32$	$\bar{v}_4 = \bar{v}_6 \cdot (-1)$	$= -0.64$
$v_7 = v_5^2$	$= 0.61$	$\bar{v}_5 = \bar{v}_7 \cdot \frac{\partial v_7}{\partial v_5} = \bar{v}_7 \cdot 2v_5$	$= -1.55$
$v_8 = v_6^2$	$= 0.1$	$\bar{v}_6 = \bar{v}_8 \cdot \frac{\partial v_8}{\partial v_6} = \bar{v}_8 \cdot 2v_6$	$= 0.64$
$v_9 = v_7 + v_8$	$= 0.71$	$\bar{v}_8 = \bar{v}_9 \cdot \frac{\partial v_9}{\partial v_8} = 1 \cdot 1$	$= 1$
		$\bar{v}_7 = \bar{v}_9 \cdot \frac{\partial v_9}{\partial v_7} = 1 \cdot 1$	$= 1$
$L_2 = v_9$	$= 0.71$	$\bar{v}_9 = \frac{\partial L_2}{\partial v_9}$	$= 1$

Remarks:

- For brevity, in the example, we assumed that the derivative $\frac{\partial}{\partial z}\sigma(z) = \sigma(z) \cdot (1 - \sigma(z))$ is already known. We could also have decomposed $\sigma(z) = \frac{1}{1+\text{exp}(-z)}$ into e.g., $v_1 = -z$, $v_2 = \text{exp}(v_1)$, $v_3 = 1 + v_2$, $v_4 = \frac{1}{v_3}$. In this case, only the four atomic derivatives would need to be known.
- The function to be automatically differentiated need not have a closed-form representation; it only has to be composed of computable and differentiable atomic steps. Thus, AD can also compute derivatives for various algorithms that may take different branches depending on the input.

Automatic Gradient Computation

Algorithm for Reverse-mode Automatic Differentiation of Scalar-valued Functions

Algorithm:	autodiff	Reverse-mode automatic differentiation
Input:	$f : \mathbf{R}^p \rightarrow \mathbf{R}$	Function to differentiate.
	$(w_1, \ldots, w_p)^T$	Point at which the gradient should be evaluated
Output:	$(\bar{w}_1, \ldots, \bar{w}_p)^T$	Gradient of f at the point $(w_1, \ldots, w_p)^T$.

1. $\bar{w}_i = 0$ `for` i `in` $1 \ldots p$ `// initialize gradients`
2. $v_1, \ldots, v_k$ = *operands*(f)
3. $\frac{\partial f}{\partial v_1}, \ldots, \frac{\partial f}{\partial v_k}$ = *gradients*(f) `// gradient of` f `wrt. its immediate operands`
4. `FOREACH` $j = 1, \ldots, k$ `DO`
5.   `IF` $v_j \in \{w_1, \ldots, w_p\}$ `THEN`
6. $\bar{v}_j$ `+=` $\frac{\partial f}{\partial v_j}$
7.   `ELSE`
8. $(\bar{w}_1, \ldots, \bar{w}_p)^T$ `+=` $\frac{\partial f}{\partial v_j} \cdot$ *autodiff*$(v_j, (w_1, \ldots, w_p)^T)$
9. `RETURN` $(\bar{w}_1, \ldots, \bar{w}_p)^T$

Remarks:

- There exists also a forward mode of automatic differentiation. One key difference is in the runtime complexity; for a function $f : \mathbf{R}^n \to \mathbf{R}^m$, to compute all $n \cdot m$ partial derivatives in the Jacobian matrix requires $O(n)$ iterations in forward mode and $O(m)$ iterations in reverse mode. Reverse mode is usually preferred in machine learning, where we typically have $m = 1$ (a scalar loss), and n arbitrarily large (e.g., billions of parameters of a deep neural network). See also [Baydin et al., 2018]