

# All of Backpropagation in Two Pages

Karl Stratos

You need to understand the chain rule (Appendix B-E) and DAGs (Appendix F) before understanding backpropagation.

## 1 Computation Graph

A computation graph is a DAG  $G = (V, A)$  in which every node  $i \in V$  is equipped with, without loss of generality, a vector-valued variable  $x^i$  of length  $d^i$ . Each non-input node  $i \in V_N$  is additionally equipped with a function

$$f^i : \prod_{j \in \text{pa}(i)} \mathbb{R}^{d^j} \rightarrow \mathbb{R}^{d^i}$$

The variables are populated as follows.

- An input node  $i \in V_I$  expects a vector  $a^i \in \mathbb{R}^{d^i}$  and populates  $x^i = a^i$ .
- A non-input node  $i \in V_N$  recursively populates  $x^i = a^i$  where

$$a^i := f^i \left( (x^j)_{j \in \text{pa}(i)} \right)$$

For convenience, we will define

$$\begin{aligned} x_I &:= (x^i)_{i \in V_I} & a_I &:= (a^i)_{i \in V_I} \\ x_I^i &:= (x^j)_{j \in \text{pa}(i)} & a_I^i &:= (a^j)_{j \in \text{pa}(i)} \quad \forall i \in V \end{aligned}$$

Thus the variable  $x^i$  at each node is a global function of  $x_I$  evaluated at  $a_I$ ; it is a local function of  $x_I^i$  evaluated at  $a_I^i$ .

## 2 Setting

We assume that the graph is connected and has an output node  $\omega \in V$  such that  $d^\omega = 1$ . Then we can view the entire graph as a scalar-valued function of  $x_I$ ,

$$\mathcal{L}^\omega : \prod_{i \in V_I} \mathbb{R}^{d^i} \rightarrow \mathbb{R}$$

where  $\mathcal{L}^\omega(x_I) := x^\omega$ . The output value  $\mathcal{L}^\omega(a_I) = a^\omega$  can be computed in runtime linear in  $|A|$  with the forward algorithm in Appendix H:

$$(a^\omega, \pi) \leftarrow \text{forward}(G, \omega, a_I)$$

where  $\pi \in \Pi_G$  is a topological ordering on  $G$  that represents the order of nodes used in computation. In particular, this populates  $x^i = a^i$  for all  $i \in V$ .

### 3 Backpropagation

The goal is to calculate the gradient of  $\mathcal{L}^\omega$ , evaluated at  $x_I = a_I$ , with respect to  $x^i$  for every  $i \neq \omega$ :

$$z^i := \left. \frac{d\mathcal{L}^\omega(x_I)}{dx^i} \right|_{x_I=a_I} \in \mathbb{R}^{1 \times d^i} \quad (1)$$

In light of the chain rule (14), this is just

$$z^i = \sum_{j \in \text{ch}(i)} \underbrace{\left. \frac{d\mathcal{L}^\omega(x_I)}{dx^j} \right|_{x_I=a_I}}_{1 \times d^j} \underbrace{\left. \frac{dx^j}{dx^i} \right|_{x_I^j=a_I^j}}_{d^j \times d^i}$$

The first key observation is that the second term in the sum is simply the Jacobian of  $f^j$ , evaluated at  $x_I^j = a_I^j$ , with respect to  $x_i$ . But because  $i \in \mathbf{pa}(j)$ , this can be analytically computed. For instance, if  $f^j = \mathbf{cmult}$  where  $\mathbf{cmult} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$  is defined as  $\mathbf{cmult}(x, x') := x \odot x'$  (Appendix G), then the Jacobian of  $\mathbf{cmult}$ , evaluated at  $(x, x') = (a, a')$ , with respect to  $x$  is

$$\left. \frac{d(x \odot x')}{dx} \right|_{(x, x')=(a, a')} = \text{diag}(a') \in \mathbb{R}^{d \times d}$$

The second key observation is that the first term in the sum is  $z^j$ , which can be recursively computed. In the base case  $j \in \mathbf{pa}(\omega)$ , this value is

$$\left. \frac{dx^\omega}{dx^j} \right|_{x_I^\omega=a_I^\omega} \in \mathbb{R}^{1 \times d^j}$$

and can again be analytically computed. The following “backpropagation” procedure computes the value of  $z^i$  for every  $i \neq \omega$  in runtime linear in  $|A|$ .

**backpropagation**  
**Input:** computation graph  $G = (V, A)$  in which  $x^i = a^i$  is populated for all  $i \in V$ , output node  $\omega \in V$ , topological ordering  $\pi \in \Pi_G$

- For  $k = |V| - 1 \dots 1$ ,
  - Set  $i = \pi(k)$ .
  - If  $i \in \mathbf{pa}(\omega)$ , populate
 
$$z^i \leftarrow \left. \frac{dx^\omega}{dx^i} \right|_{x_I^\omega=a_I^\omega}$$
  - If  $i \notin \mathbf{pa}(\omega)$ , populate
 
$$z^i \leftarrow \sum_{j \in \text{ch}(i)} z^j \left. \frac{dx^j}{dx^i} \right|_{x_I=a_I}$$

Another way to view the algorithm is a sum-product algorithm on a DAG (Appendix F.2) since

$$\left. \frac{d\mathcal{L}^\omega(x_I)}{dx^i} \right|_{x_I=a_I} = \sum_{(i_1 \dots i_n) \in P(i, \omega)} \left. \frac{dx^\omega}{dx^{i_{n-1}}} \right|_{x_I^\omega=a_I^\omega} \left. \frac{dx^{i_{n-1}}}{dx^{i_{n-2}}} \right|_{x_I^{i_{n-1}}=a_I^{i_{n-1}}} \dots \left. \frac{dx^{i_2}}{dx^{i_1}} \right|_{x_I^{i_1}=a_I^{i_1}}$$

But this view is not necessary to see the correctness of the algorithm.

## A Notation

The set of unit vectors in  $\mathbb{R}^n$  is denoted by  $\mathcal{S}^n := \{v \in \mathbb{R}^n : \|v\|_2 = 1\}$ . The  $i$ -th standard basis vector in  $\mathbb{R}^n$  is denoted by  $e_i \in \{0, 1\}^n$ . The norm of a vector  $x \in \mathbb{R}^n$  is denoted by  $\|x\|$ : we assume a fixed choice of  $\|\cdot\|$  (e.g., Euclidean), but we make no assumption about the choice. The  $(n-1)$ -dimensional probability simplex is denoted by  $\Delta^{n-1} := \{v \in \mathbb{R}^n : v \geq 0, \|v\|_1 = 1\}$ . The component-wise multiplication of vectors  $x, x' \in \mathbb{R}^n$  is denoted by  $x \odot x' \in \mathbb{R}^n$ . The concatenation of vectors  $x \in \mathbb{R}^n$  and  $x' \in \mathbb{R}^{n'}$  is denoted by  $x \oplus x' \in \mathbb{R}^{n+n'}$ . The sigmoid function is defined as  $\sigma(x) := (1 + \exp(-x))^{-1}$ . We write  $I_{n \times n}$  to denote the  $n \times n$  identity matrix,  $0_{m \times n}$  to denote the  $m \times n$  zero matrix,  $1_n$  and  $0_n$  to denote the  $n$ -dimensional vector of ones and zeros. Given a vector  $v \in \mathbb{R}^n$ ,  $\text{diag}(v) \in \mathbb{R}^{n \times n}$  refers to a diagonal matrix with  $[\text{diag}(v)]_{i,i} = v_i$ .

## B Scalar-Valued Function of a Scalar Variable

Consider  $f : \mathbb{R} \rightarrow \mathbb{R}$  and  $a \in \mathbb{R}$ .

### B.1 Limit

The **limit of  $f(x)$  as  $x$  approaches  $a$**  is a constant  $L \in \mathbb{R}$  satisfying the following: given any  $\epsilon > 0$ , we can find  $\delta > 0$  such that if  $x \in \mathbb{R}$  satisfies  $|x - a| < \delta$ , then  $|f(x) - L| < \epsilon$ . In this case, we say the limit exists and write

$$\lim_{x \rightarrow a} f(x) = L \tag{2}$$

**Theorem B.1.** *If the limit of  $f(x)$  as  $x$  approaches  $a$  exists, then it is unique.*

$f$  is **continuous at  $a$**  if  $f(a) = \lim_{x \rightarrow a} f(x)$ . Note that  $f$  may not be continuous but still have a limit at  $a$ .

### B.2 Derivative

The **derivative of  $f$  at  $a$**  is a unique scalar  $f'(a) \in \mathbb{R}$  such that

$$\lim_{x \rightarrow a} \frac{f(x) - (f(a) + f'(a)(x - a))}{x - a} = 0 \tag{3}$$

This definition is equivalent to

$$f'(a) := \lim_{\epsilon \rightarrow 0} \frac{f(a + \epsilon) - f(a)}{\epsilon} \tag{4}$$

We say  $f$  is **differentiable at  $a$**  if  $f'(a)$  exists.

### B.3 Chain Rule

We write

$$\left. \frac{df(x)}{dx} \right|_{x=a} \in \mathbb{R}$$

to mean “the derivative of  $f : \mathbb{R} \rightarrow \mathbb{R}$  with respect to parameter  $x$  when  $x = a$ ”. This is of course just  $f'(a)$ , but what if we introduce  $g : \mathbb{R} \rightarrow \mathbb{R}$  and want to compute

$$\left. \frac{dg(f(x))}{dx} \right|_{x=a} \in \mathbb{R}$$

The central tool for this problem is the **chain rule**

$$\left. \frac{dg(f(x))}{dx} \right|_{x=a} = \left. \frac{dg(y)}{dy} \right|_{y=f(a)} \times \left. \frac{df(x)}{dx} \right|_{x=a} \quad (5)$$

which can now be calculated as  $g'(f(a)) \times f'(a)$ . Why is this true? A non-rigorous but illuminating argument is as follows. By the definition of the derivative (3)

$$\begin{aligned} g(y) &\approx g(b) + g'(b)(y - b) & \forall y, b \in \mathbb{R} \\ f(x) &\approx f(a) + f'(a)(x - a) & \forall x, a \in \mathbb{R} \end{aligned}$$

Use  $y = f(x)$  and  $b = f(a)$ , and expand  $f(x)$  by its linear approximation to have

$$g(f(x)) \approx g(f(a)) + g'(f(a))f'(a)(x - a) \quad \forall x, b \in \mathbb{R}$$

This means that  $g'(f(a))f'(a)$  is the derivative of  $g(f(x))$  with respect to  $x$ .

## C Scalar-Valued Function of a Vector Variable

Consider  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  and  $a \in \mathbb{R}^n$ .

### C.1 Limit

The limit of a function of a vector variable is straightforward to generalize from the scalar-variable case. The **limit of  $f(x)$  as  $x$  approaches  $a$**  is a constant  $L \in \mathbb{R}$  satisfying the following: given any  $\epsilon > 0$ , we can find  $\delta > 0$  such that if  $x \in \mathbb{R}^n$  satisfies  $\|x - a\| < \delta$ , then  $|f(x) - L| < \epsilon$ . The uniqueness and continuity are derived similarly.

### C.2 Directional/Partial Derivative

The **directional derivative of  $f$  at  $a$  in the direction of  $v \in \mathcal{S}^n$**  is

$$D_v f(a) := f'_v(0) = \lim_{\epsilon \rightarrow 0} \frac{f(a + \epsilon v) - f(a)}{\epsilon} \quad (6)$$

where  $f_v : \mathbb{R} \rightarrow \mathbb{R}$  is defined by  $f_v(t) := f(a + tv)$ . This is a natural reduction to the scalar-variable derivative (4) (equivalent when  $n = 1$ ). The  **$i$ -th partial derivative of  $f$  at  $a$**  is simply the directional derivative in the direction of  $e_i$ :

$$\frac{\partial f(a)}{\partial x_i} := D_{e_i} f(a) = \lim_{\epsilon \rightarrow 0} \frac{f(a_1, \dots, a_i + \epsilon, \dots, a_n) - f(a_1, \dots, a_n)}{\epsilon}$$

### C.3 Gradient

The **gradient of  $f$  at  $a$**  is a unique vector  $\nabla f(a) \in \mathbb{R}^n$  such that

$$\lim_{x \rightarrow a} \frac{f(x) - (f(a) + \nabla f(a)^\top (x - a))}{\|x - a\|} = 0 \quad (7)$$

This is a natural generalization of the scalar-variable derivative (3) (equivalent when  $n = 1$ ). We say  $f$  is **differentiable at  $a$**  if  $\nabla f(a)$  exists.

Equivalently, the gradient of  $f$  at  $a$  is a unique vector  $\nabla f(a) \in \mathbb{R}^n$  such that

$$D_v f(a) = \nabla f(a)^\top v \quad \forall v \in \mathcal{S}^n \quad (8)$$

This version is useful because it tells us that for  $f(x)$  at  $x = a$ ,  $-\nabla f(a)$  is the direction with the maximum rate of decrease  $-\|\nabla f(a)\|^2$ ,  $\nabla f(a)$  is the direction with the maximum rate of increase  $\|\nabla f(a)\|^2$ , and any direction orthogonal to  $\nabla f(a)$  does not change the function value.

#### C.4 Gradient as Partial Derivatives

It is easy to see that if  $f$  is differentiable at  $a$ , then the gradient must have the form

$$\nabla f(a) = \left( \frac{\partial f(a)}{\partial x_1} \dots \frac{\partial f(a)}{\partial x_n} \right) \quad (9)$$

because the gradient must satisfy  $[\nabla f(a)]_i = \nabla f(a)^\top e_i = D_{e_i} f(a) = \frac{\partial f(a)}{\partial x_i}$  for all  $i \in \{1 \dots n\}$  by definition (8). However,  $f$  may *not* be differentiable at  $a$  even if all partial and directional derivatives exist at  $a$ . The following result allows us to eliminate this subtlety.

**Theorem C.1.** *If the partial derivatives of  $f$  are continuous around  $a$ , then  $f$  is differentiable at  $a$ .*

We generally only discuss functions with continuous partial derivatives (thus differentiable), so we will use (9) as a definition of the gradient.

## D Vector-Valued Function of a Vector Variable

Consider  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$  and  $a \in \mathbb{R}^n$ . We will view  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$  simply as a concatenation of  $f_1 \dots f_m : \mathbb{R}^n \rightarrow \mathbb{R}$ . That is,

$$f(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{bmatrix} \quad \forall x \in \mathbb{R}^n$$

### D.1 Total Derivative

The **total derivative of  $f$  at  $a$**  is a unique matrix  $T_a^f \in \mathbb{R}^{m \times n}$  such that

$$\lim_{x \rightarrow a} \frac{\|f(x) - (f(a) + T_a^f(x - a))\|}{\|x - a\|} = 0 \quad (10)$$

This is a natural generalization of the gradient (7) (equivalent when  $m = 1$ ): the linear function  $f(a) + T_a^f(x - a)$  is a linear approximation of  $f(x)$  around  $a$ . We say  $f$  is **differentiable at  $a$**  if  $T_a^f$  exists.

## D.2 Total Derivative as Jacobian

It is easy to see that when  $f_1 \dots f_m$  are differentiable, we have

$$T_a^f = \begin{bmatrix} \nabla f_1(a)^\top \\ \vdots \\ \nabla f_m(a)^\top \end{bmatrix} =: J_f(a) \quad (11)$$

where the matrix  $J_f(a) \in \mathbb{R}^{m \times n}$  whose  $i$ -th row is the gradient of  $f_i$  at  $a$  is called the **Jacobian of  $f$  at  $a$** . Thus we will equate the Jacobian with the total derivative. It is useful to view the Jacobian in terms of scalar derivatives: the  $(i, j)$ -th value of  $J_f(a) \in \mathbb{R}^{m \times n}$  is the derivative of  $f_i : \mathbb{R} \rightarrow \mathbb{R}$  with respect to  $x_j \in \mathbb{R}$  when  $x = a$ ,

$$[J_f(a)]_{i,j} = \left. \frac{df_i(x)}{dx_j} \right|_{x=a} \quad (12)$$

## D.3 Chain Rule

We now revisit the chain rule. We write

$$\left. \frac{df(x)}{dx} \right|_{x=a} \in \mathbb{R}^{m \times n}$$

to mean “the Jacobian of  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$  with respect to parameter  $x$  when  $x = a$ ”. This is of course just  $J_f(a)$ , but what if we introduce  $g : \mathbb{R}^m \rightarrow \mathbb{R}^d$  and want to compute

$$\left. \frac{dg(f(x))}{dx} \right|_{x=a} \in \mathbb{R}^{d \times n}$$

Beautifully, the **chain rule** takes the same form in (13):

$$\underbrace{\left. \frac{dg(f(x))}{dx} \right|_{x=a}}_{d \times n} = \underbrace{\left. \frac{dg(y)}{dy} \right|_{y=f(a)}}_{d \times m} \underbrace{\left. \frac{df(x)}{dx} \right|_{x=a}}_{m \times n} \quad (13)$$

which can now be calculated as matrix product  $J_g(f(a))J_f(a)$ . We can again convince ourselves that this is true by using the definition of the total derivative (10) to derive

$$g(f(x)) \approx g(f(a)) + J_g(f(a))J_f(a)(x - a) \quad \forall x, a \in \mathbb{R}$$

This means that  $J_g(f(a))J_f(a)$  is the total derivative of  $g(f(x))$  with respect to  $x$ .

**Sum over derivatives.** In scalar form, the chain rule states that the derivative of  $g_i(f(x)) \in \mathbb{R}$  with respect to  $x_j \in \mathbb{R}$  is

$$\left. \frac{dg_i(f(x))}{dx_j} \right|_{x=a} = \sum_{k=1}^m \left. \frac{dg_i(y)}{dy_k} \right|_{y=f(a)} \times \left. \frac{df_k(x)}{dx_j} \right|_{x=a}$$

This is almost the same as the scalar-variable chain rule (5) except that we sum over partial contributions from  $x_j$  through  $m$  arguments  $y_k = f_k(x)$  in  $g(y_1 \dots y_m)$ .

**Sum over Jacobians.** Sometimes it is useful to view  $g : \mathbb{R}^m \rightarrow \mathbb{R}^d$  as a function of multiple vectors instead of one. Let  $R_1 \dots R_K$  be any  $K$ -partition of indices  $\{1 \dots m\}$  and define  $f^{(k)} : \mathbb{R}^n \rightarrow \mathbb{R}^{|R_k|}$  by  $f^{(k)}(x) = (f_i(x))_{i \in R_k}$ . We now view  $g$  as a function

$$g : \mathbb{R}^{|R_1|} \times \dots \times \mathbb{R}^{|R_K|} \rightarrow \mathbb{R}^d$$

that takes  $K$  input vectors  $y^{(1)} \dots y^{(K)}$  where  $y^{(k)} = f^{(k)}(x)$ . The chain rule states that

$$\underbrace{\frac{dg(f(x))}{dx} \Big|_{x=a}}_{d \times n} = \sum_{k=1}^K \underbrace{\frac{dg(y^{(1)}, \dots, y^{(K)})}{dy^{(k)}} \Big|_{y=f(a)}}_{d \times |R_k|} \underbrace{\frac{df^{(k)}(x)}{dx} \Big|_{x=a}}_{|R_k| \times n} \quad (14)$$

where  $y = f(a)$  means  $y^{(k)} = f^{(k)}(a)$  for all  $k \in \{1 \dots K\}$ . If  $K = 1$ , we recover the single-vector case (13).

## E Tensor-Valued Function of a Tensor Variable

Now that we have covered the case of a vector-valued function of a vector variable, we can easily extend it to the general case of

$$f : \mathbb{R}^{n_1 \times \dots \times n_N} \rightarrow \mathbb{R}^{m_1 \times \dots \times m_M}$$

with input tensor  $A \in \mathbb{R}^{n_1 \times \dots \times n_N}$ . This is achieved by “vectorizing” the tensor. For example,  $A$  is viewed as a vector of length  $(n_1 \dots n_N)$  whose indices

$$i \in \{1 \dots (n_1 \dots n_N)\}$$

are in one-to-one correspondence with tuples

$$(i_1, \dots, i_M) \in \{1 \dots n_1\} \times \dots \times \{1 \dots n_M\}$$

Let  $\mathbf{ind}(i_1, \dots, i_M)$  denote vector index corresponding to the tensor index tuple  $(i_1, \dots, i_M)$ . Then the total derivative of  $f$  at  $A$  is viewed as a “matrix” of dimensions  $(m_1 \dots m_M) \times (n_1 \dots n_N)$  with elements

$$\left[ \frac{df(X)}{dX} \Big|_{X=A} \right]_{\mathbf{ind}(i_1, \dots, i_M), \mathbf{ind}(j_1, \dots, j_N)} = \frac{df_{i_1, \dots, i_M}(X)}{dX_{j_1, \dots, j_N}} \Big|_{X=A} \quad (15)$$

**Chain rule.** Suppose we introduce

$$g : \mathbb{R}^{m_1 \times \dots \times m_M} \rightarrow \mathbb{R}^{d_1 \times \dots \times d_D}$$

and want to compute the total derivative of  $g(f(X))$  with respect to  $X$  at  $A$ . Again taking the vectorized view, we can invoke the chain rule in (13) and calculate

$$\underbrace{\frac{dg(f(X))}{dX} \Big|_{X=A}}_{(d_1 \dots d_D) \times (n_1 \dots n_N)} = \underbrace{\frac{dg(y)}{dy} \Big|_{y=f(A)}}_{(d_1 \dots d_D) \times (m_1 \dots m_M)} \underbrace{\frac{df(X)}{dX} \Big|_{X=A}}_{(m_1 \dots m_M) \times (n_1 \dots n_N)}$$

Equivalently, the chain rule states that the total derivative is a  $(D + N)$ -th-order tensor of dimensions  $d_1 \times \dots \times d_D \times n_1 \times \dots \times n_N$  whose  $(i_1, \dots, i_D, j_1, \dots, j_N)$ -th element is

$$\frac{dg_{i_1, \dots, i_D}(f(X))}{dX_{j_1, \dots, j_N}} \Big|_{X=A} = \sum_{k_1=1}^{m_1} \dots \sum_{k_M=1}^{m_M} \frac{dg_{i_1, \dots, i_D}(B)}{dB_{k_1, \dots, k_M}} \Big|_{B=f(A)} \times \frac{df_{k_1, \dots, k_M}(X)}{dX_{j_1, \dots, j_N}} \Big|_{X=A}$$

## F Directed Acyclic Graph (DAG)

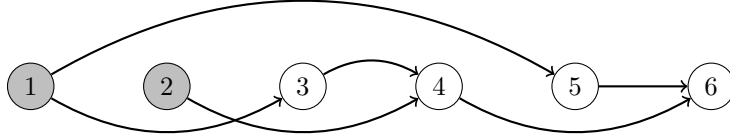
### F.1 Terminology

A **directed graph** is a pair  $G = (V, A)$  where  $V = \{1 \dots |V|\}$  is a set of **nodes** and  $A \in V \times V$  is a set of directed **arcs**. We sometimes denote the head and tail of an arc  $a = (i, j)$  by  $a^h = i$  and  $a^t = j$ . A **directed acyclic graph (DAG)** is a directed graph with no cycles. Equivalently, a DAG is a directed graph with a **topological ordering**: a sequence  $\pi$  of  $V$  such that for every arc  $(i, j) \in A$ ,  $i$  comes before  $j$  in  $\pi$ . Let  $\Pi_G$  denote the set of all topological orderings in  $G$ .

Given a node  $i \in V$ , we denote the set of its parents by  $\mathbf{pa}(i) := \{j \in V : (j, i) \in A\}$  and the set of its children by  $\mathbf{ch}(i) := \{j \in V : (i, j) \in A\}$ . We say  $i \in V$  is a **input node** if  $\mathbf{pa}(i) = \emptyset$ . Let  $V_I$  and  $V_N$  denote the set of input and non-input nodes: together, they form a partition of  $V$ . We say  $i \in V$  is an **output node** if  $\mathbf{ch}(i) = \emptyset$ . The set of **paths** from  $i \in V$  to  $j \in V$  where  $i \neq j$  is

$$P(i, j) := \{(a_1 \dots a_n) \in A^n : n \geq 2, a_1^h = i, a_n^t = j, a_{k-1}^t = a_k^h \forall k = 2 \dots n\}$$

Denote the set of nodes that can reach  $j \in V$  by  $\rho(j) := \{i \in V : |P(i, j)| \geq 1\}$ . Here is an example of a DAG (input nodes shaded for readability):



$$V = \{1, 2, 3, 4, 5, 6\}$$

$$A = \{(1, 3), (1, 5), (2, 4), (3, 4), (4, 6), (5, 6)\}$$

$$\mathbf{pa}(4) = \{2, 3\}$$

$$\mathbf{ch}(1) = \{3, 5\}$$

$$\Pi_G = \{(1, 2, 3, 4, 5, 6), (2, 1, 3, 4, 5, 6)\}$$

$$V_I = \{1, 2\}$$

$$V_N = \{3, 4, 5, 6\}$$

$$P(1, 6) = \{((1, 3), (3, 4), (4, 6)), ((1, 5), (5, 6))\}$$

$$P(2, 6) = \{((2, 4), (4, 6))\}$$

$$P(2, 5) = \emptyset$$

$$\rho(6) = \{1, 2, 3, 4, 5\}$$

$$\rho(5) = \{1\}$$

$$\rho(4) = \{1, 2, 3\}$$

### F.2 Sum-Product Algorithm on DAGs

Let  $\mathcal{Q}$  be any set equipped with associative binary operations  $+$  and  $*$ . We assume that the multiplicative operation  $*$  is distributive over  $+$ . We assume that the additive operation  $+$  is commutative but the multiplicative operation  $*$  may not be. For



instance,  $\mathcal{Q}$  can be the set of matrices and  $(+, *)$  can be the matrix addition and multiplication (applicable only to matrices with correct dimensions).

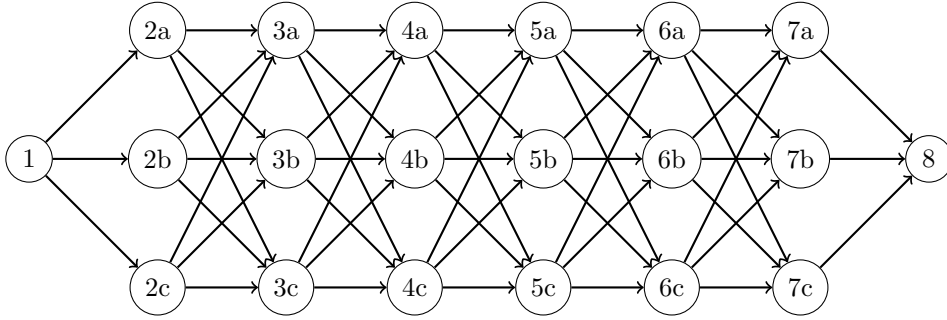
Suppose we have a DAG  $G = (V, A)$  in which each arc  $(i, j) \in A$  is associated with  $Q^{i \rightarrow j} \in \mathcal{Q}$ . A computation of interest is: given the last node  $t \in V$  in a topological ordering  $\pi$ , calculate

$$\mu(s) := \sum_{(a_1 \dots a_n) \in P(s, t)} \left( Q^{a_n \rightarrow a_n^t} * \dots * Q^{a_1 \rightarrow a_1^t} \right) \quad (16)$$

for all reachable  $s \in \rho(t)$ . Note the reverse order of multiplication: because  $*$  is not commutative, it will be important to respect this order. For instance, in the above example DAG with  $t = 6$ , we have

$$\mu(1) = (Q^{4 \rightarrow 6} * Q^{3 \rightarrow 4} * Q^{1 \rightarrow 3}) + (Q^{5 \rightarrow 6} * Q^{1 \rightarrow 5})$$

Explicitly summing over all paths is not a good idea since the number of paths in  $P(s, t)$  may grow exponentially in the length of a path. For instance, in the following DAG



the number of paths in  $P(1, 8)$  is  $3^6 = 729$ . However, observe that:

- If  $s \in \mathbf{pa}(t)$ :

$$\mu(s) = Q^{s \rightarrow t}$$

- If  $s \notin \mathbf{pa}(t)$ :

$$\begin{aligned} \mu(s) &= \sum_{(a_1 \dots a_n) \in P(s, t)} \left( Q^{a_n \rightarrow a_n^t} * \dots * Q^{a_2 \rightarrow a_2^t} * Q^{a_1 \rightarrow a_1^t} \right) \\ &= \sum_{i \in \mathbf{ch}(s)} \sum_{(a_2 \dots a_n) \in P(i, t)} \left( Q^{a_n \rightarrow a_n^t} * \dots * Q^{a_2 \rightarrow a_2^t} * Q^{s \rightarrow i} \right) \\ &= \sum_{i \in \mathbf{ch}(s)} \left( \sum_{(a_2 \dots a_n) \in P(i, t)} Q^{a_n \rightarrow a_n^t} * \dots * Q^{a_2 \rightarrow a_2^t} \right) * Q^{s \rightarrow i} \\ &= \sum_{i \in \mathbf{ch}(s)} \mu(i) * Q^{s \rightarrow i} \end{aligned}$$

where the third equality uses the distributivity of  $*$  over  $+$ .

Thus we can use the following one-liner dynamic programming algorithm.

**Input:**  $G = (V, A)$ , topological ordering  $\pi \in \Pi_G$ ,  $t = \pi(|V|)$

**Output:**  $\mu(s)$  in (16) for all  $s \in \rho(t)$

- For  $i = |V| - 1 \dots 1$ , set  $s = \pi(i)$  and compute

$$\mu(s) = \begin{cases} Q^{s \rightarrow t} & \text{if } s \in \mathbf{pa}(t) \\ \sum_{j \in \mathbf{ch}(s)} \mu(j) * Q^{s \rightarrow j} & \text{otherwise} \end{cases}$$

It is critical to follow a reverse topological ordering since it guarantees that  $\mu(j)$  is computed for all children  $j$  of  $s$  before  $\mu(s)$  is computed. The number of computation steps is  $|A|$ : in the example above, it is  $51 \ll 729$ .

## G Example Functions in a Computation Graph

### G.1 Common Functions

<b>add</b> : $\mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$	<b>add</b> ( $x, x'$ ) := $x + x'$	
<b>cmult</b> : $\mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$	<b>cmult</b> ( $x, x'$ ) := $x \odot x'$	
<b>concat</b> : $\mathbb{R}^d \times \mathbb{R}^{d'} \rightarrow \mathbb{R}^{d+d'}$	<b>concat</b> ( $x, x'$ ) := $x \oplus x'$	
<b>mvmult</b> : $\mathbb{R}^{d' \times d} \times \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$	<b>mvmult</b> ( $W, x$ ) := $Wx$	
<b>pick</b> : $\mathbb{R}^d \times \{1 \dots d\} \rightarrow \mathbb{R}$	<b>pick</b> ( $x, i$ ) := $x_i$	$\forall i = 1 \dots d$
<b>power</b> : $\mathbb{R}^d \times \mathbb{Z} \rightarrow \mathbb{R}^d$	<b>power</b> <sub><math>i</math></sub> ( $x, n$ ) := $x_i^n$	$\forall i = 1 \dots d$
<b>tanh</b> : $\mathbb{R}^d \rightarrow \mathbb{R}^d$	<b>tanh</b> <sub><math>i</math></sub> ( $x$ ) := $\tanh(x_i)$	$\forall i = 1 \dots d$
<b>sigmoid</b> : $\mathbb{R}^d \rightarrow \mathbb{R}^d$	<b>sigmoid</b> <sub><math>i</math></sub> ( $x$ ) := $\sigma(x_i)$	$\forall i = 1 \dots d$
<b>softmax</b> : $\mathbb{R}^d \rightarrow \mathbb{R}^d$	<b>softmax</b> <sub><math>i</math></sub> ( $x$ ) := $\frac{\exp(x_i)}{\sum_{j=1}^d \exp(x_j)}$	$\forall i = 1 \dots d$

### G.2 Jacobians

Multi-argument functions.

<b>add</b>	$\frac{d(x + x')}{dx} = I_{d \times d}$	$\frac{d(x + x')}{dx'} = I_{d \times d}$
<b>cmult</b>	$\frac{d(x \odot x')}{dx} = \text{diag}(x')$	$\frac{d(x \odot x')}{dx'} = \text{diag}(x)$
<b>concat</b>	$\frac{d(x \oplus x')}{dx} = \begin{bmatrix} I_{d \times d} \\ 0_{d' \times d} \end{bmatrix}$	$\frac{d(x \oplus x')}{dx'} = \begin{bmatrix} 0_{d \times d'} \\ I_{d' \times d'} \end{bmatrix}$
<b>mvmult</b>	$\frac{d[Wx]_i}{dW_{j,k}} = \begin{cases} x_k & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$	$\frac{d(Wx)}{dx} = W$
<b>pick</b>	$\frac{d\mathbf{pick}(x, i)}{dx} = e_i$	
<b>power</b>	$\frac{d\mathbf{power}(x, n)}{dx} = n \times \mathbf{power}(x, n - 1)$	

Single-argument functions.

$$\begin{aligned}
 \text{tanh} \quad \frac{d\text{tanh}_i(x)}{dx_j} &= \begin{cases} 1 - \tanh(x_i)^2 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \\
 \text{sigmoid} \quad \frac{d\text{sigmoid}(x)_i}{dx_j} &= \begin{cases} \sigma(x_i) \times (1 - \sigma(x_i)) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \\
 \text{softmax} \quad \frac{d\text{softmax}_i(x)}{dx_j} &= \begin{cases} \text{softmax}_i(x) \times (1 - \text{softmax}_i(x)) & \text{if } i = j \\ -\text{softmax}_i(x) \times \text{softmax}_j(x) & \text{otherwise} \end{cases}
 \end{aligned}$$

## H Forward Computation

**forward**

**Input:** computation graph  $G = (V, A)$ , output node  $\omega \in V$ , input value  $a_I$

**Output:**  $\mathcal{L}^\omega(x_I) := x^\omega$  evaluated at  $a_I$ , topological ordering  $\pi \in \Pi_G$

- $a^\omega \leftarrow \text{forward-rec}(G, \omega, a_I, \pi \leftarrow ())$
- Return  $(a^\omega, \pi)$ .

**forward-rec**

**Input:** computation graph  $G = (V, A)$ ,  $i \in V$ , input value  $a_I$ , topological ordering in construction  $\pi$

- If  $i \in V_I$ , set  $\pi \leftarrow \pi + (i)$  and return  $x^i \leftarrow a^i$ .
- Otherwise,
  - Calculate  $a^j \leftarrow \text{forward-rec}(G, a_I, j, \pi)$  for each  $j \in \text{pa}(i)$ .
  - Set  $\pi \leftarrow \pi + (i)$  and return  $x^i \leftarrow f^i \left( (a^j)_{j \in \text{pa}(i)} \right)$ .