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

$$f^i : \bigtimes_{j \in \mathbf{pa}(i)} \mathbb{R}^{d^j} \to \mathbb{R}^{d^i}$$

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

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

For convenience, we will define

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

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,

$$\mathcal{L}^\omega : \bigtimes_{i \in V_I} \mathbb{R}^{d^i} \to \mathbb{R}$$

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 G:

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

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 $i \in V$:

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

In light of the chain rule (15), this is just

$$z^i = \sum_{j \in \mathbf{ch}(i)} \underbrace{\left.\frac{d\mathcal{L}^\omega(x_I)}{dx^j}\right|_{x_I = a_I}}_{1 \times d^j} \underbrace{\left.\frac{dx^j}{dx^i}\right|_{x_I^j = a_I^j}}_{d^j \times d^i} \tag{2}$$

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 $\mathbf{cmult} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ is defined as $\mathbf{cmult}(x, x') := x \odot x'$ (Appendix H), then the Jacobian of $\mathbf{cmult}$, evaluated at $(x, x') = (a, a')$, with respect to x is

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

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

$$\left.\frac{d\mathcal{L}^\omega(x_I)}{dx^\omega}\right|_{x_I = a_I} = \left.\frac{dx^\omega}{dx^\omega}\right|_{x_I^\omega = a_I^\omega} = 1$$

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 $i \in V$, topological ordering $\pi \in \Pi_G$

- Set $\omega = \pi(|V|)$ and initialize $z^\omega = 1$.
- For $k = |V| - 1 \ldots 1$,
  - Set $i = \pi(k)$ and compute

$$z^i \leftarrow \sum_{j \in \mathbf{ch}(i)} z^j \left.\frac{dx^j}{dx^i}\right|_{x_I = a_I}$$

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

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

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 $v \in \mathbb{R}^n$, $\mathrm{diag}(v) \in \mathbb{R}^{n \times n}$ refers to a diagonal matrix with $[\mathrm{diag}(v)]_{i,i} = v_i$.

B Scalar-Valued Function of a Scalar Variable

Consider $f : \mathbb{R} \to \mathbb{R}$ 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

$$\lim_{x \to a} f(x) = L \tag{3}$$

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

f is **continuous at** a if $f(a) = \lim_{x \to a} f(x)$. 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

$$\lim_{x \to a} \frac{f(x) - (f(a) + f'(a)(x - a))}{x - a} = 0 \tag{4}$$

This definition is equivalent to

$$f'(a) := \lim_{\epsilon \to 0} \frac{f(a + \epsilon) - f(a)}{\epsilon} \tag{5}$$

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 $f : \mathbb{R} \to \mathbb{R}$ with respect to parameter x when $x = a$". This is of course just $f'(a)$, but what if we introduce $g : \mathbb{R} \to \mathbb{R}$ 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**

$$\left.\frac{dg(f(x))}{dx}\right|_{x=a} = \left.\frac{dg(y)}{dy}\right|_{y=f(a)} \times \left.\frac{df(x)}{dx}\right|_{x=a} \tag{6}$$

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 (4)

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

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

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

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

C Scalar-Valued Function of a Vector Variable

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

C.1 Limit

The limit of a function of a vector variable is straightforward to generalizes 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

$$D_v f(a) := f_v'(0) = \lim_{\epsilon \to 0} \frac{f(a + \epsilon v) - f(a)}{\epsilon} \tag{7}$$

where $f_v : \mathbb{R} \to \mathbb{R}$ is defined by $f_v(t) := f(a + tv)$. This is a natural reduction to the scalar-variable derivative (5) (equivalent when $n = 1$). The **i-th partial derivative of f at a** is simply the directional derivative in the direction of e_i:

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

C.3 Gradient

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

$$\lim_{x \to a} \frac{f(x) - (f(a) + \nabla f(a)^\top (x - a))}{||x - a||} = 0 \tag{8}$$

This is a natural generalization of the scalar-variable derivative (4) (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

$$D_v f(a) = \nabla f(a)^\top v \qquad\qquad \forall v \in \mathcal{S}^n \tag{9}$$

This version is useful because it tells us that for $f(x)$ at $x = a$, $-\nabla f(a)/\left|\left|\nabla f(a)\right|\right|$ is the direction with the maximum rate of decrease $-\left|\left|\nabla f(a)\right|\right|^2$, $\nabla f(a)/\left|\left|\nabla f(a)\right|\right|$ is the direction with the maximum rate of increase $\left|\left|\nabla f(a)\right|\right|^2$, 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

$$\nabla f(a) = \left(\frac{\partial f(a)}{\partial x_1} \cdots \frac{\partial f(a)}{\partial x_n}\right) \tag{10}$$

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 (9). 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 (10) as a definition of the gradient.

D Vector-Valued Function of a Vector Variable

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

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

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

$$\lim_{x \to a} \frac{\left|\left|f(x) - (f(a) + T_a^f(x-a))\right|\right|}{\left|\left|x - a\right|\right|} = 0 \tag{11}$$

This is a natural generalization of the gradient (8) (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

$$T_a^f = \begin{bmatrix} \nabla f_1(a)^\top \\ \vdots \\ \nabla f_m(a)^\top \end{bmatrix} =: J_f(a) \tag{12}$$

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 $f_i : \mathbb{R} \to \mathbb{R}$ with respect to $x_j \in \mathbb{R}$ when $x = a$,

$$[J_f(a)]_{i,j} = \left.\frac{df_i(x)}{dx_j}\right|_{x=a} \tag{13}$$

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 $f : \mathbb{R}^n \to \mathbb{R}^m$ with respect to parameter x when $x = a$". This is of course just $J_f(a)$, but what if we introduce $g : \mathbb{R}^m \to \mathbb{R}^d$ 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 (14):

$$\underbrace{\left.\frac{dg(f(x))}{dx}\right|_{x=a}}_{d \times n} = \underbrace{\left.\frac{dg(y)}{dy}\right|_{y=f(a)}}_{d \times m} \underbrace{\left.\frac{df(x)}{dx}\right|_{x=a}}_{m \times n} \tag{14}$$

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 (11) to derive

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

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 (6) 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 $g : \mathbb{R}^m \to \mathbb{R}^d$ 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 $f^{(k)} : \mathbb{R}^n \to \mathbb{R}^{|R_k|}$ by $f^{(k)}(x) = (f_i(x))_{i \in R_k}$. We now view g as a function

$$g : \mathbb{R}^{|R_1|} \times \cdots \times \mathbb{R}^{|R_K|} \to \mathbb{R}^d$$

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

$$\underbrace{\left.\frac{dg(f(x))}{dx}\right|_{x=a}}_{d\times n} = \sum_{k=1}^{K} \underbrace{\left.\frac{dg(y^{(1)}, \dots, y^{(K)})}{dy^{(k)}}\right|_{y=f(a)}}_{d\times|R_k|} \underbrace{\left.\frac{df^{(k)}(x)}{dx}\right|_{x=a}}_{|R_k|\times n} \tag{15}$$

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 (14).

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

$$f : \mathbb{R}^{n_1 \times \cdots \times n_N} \to \mathbb{R}^{m_1 \times \cdots \times m_M}$$

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

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

are in one-to-one correspondence with tuples

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

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 $(m_1 \cdots m_M) \times (n_1 \cdots n_N)$ with elements

$$\left[\left.\frac{df(X)}{dX}\right|_{X=A}\right]_{\mathbf{ind}(i_1,\dots,i_M),\mathbf{ind}(j_1,\dots,j_N)} = \left.\frac{df_{i_1,\dots,i_M}(X)}{dX_{j_1,\dots,j_N}}\right|_{X=A} \tag{16}$$

Chain rule. Suppose we introduce

$$g : \mathbb{R}^{m_1 \times \cdots \times m_M} \to \mathbb{R}^{d_1 \times \cdots \times d_D}$$

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 (14) and calculate

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

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

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

F Directed Acyclic Graph (DAG)

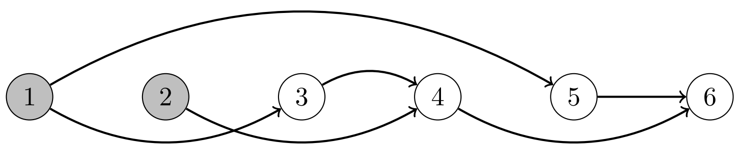
F.1 Terminology

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 acylic graph (DAG)** is a directed graph with no cycles. Equivalently, a DAG is a directed graph with a **topological ordering**: a sequence π of V such that for every arc $(i, j) \in A$, i comes before j in π. Let Π_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 $\mathbf{pa}(i) = \varnothing$. 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 $\mathbf{ch}(i) = \varnothing$. The set of **paths** from $i \in V$ to $j \in V$ where $i \neq j$ is

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

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):



$$\begin{aligned}
V &= \{1, 2, 3, 4, 5, 6\} \\
A &= \{(1, 3), (1, 5), (2, 4), (3, 4), (4, 6), (5, 6)\} \\
\mathbf{pa}(4) &= \{2, 3\} \\
\mathbf{ch}(1) &= \{3, 5\} \\
\Pi_G &= \{(1, 2, 3, 4, 5, 6),\ (2, 1, 3, 4, 5, 6)\} \\
V_I &= \{1, 2\} \\
V_N &= \{3, 4, 5, 6\} \\
P(1, 6) &= \{((1, 3), (3, 4), (4, 6)),\ ((1, 5), (5, 6))\} \\
P(2, 6) &= \{((2, 4), (4, 6))\} \\
P(2, 5) &= \varnothing \\
\rho(6) &= \{1, 2, 3, 4, 5\} \\
\rho(5) &= \{1\} \\
\rho(4) &= \{1, 2, 3\}
\end{aligned}$$

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).

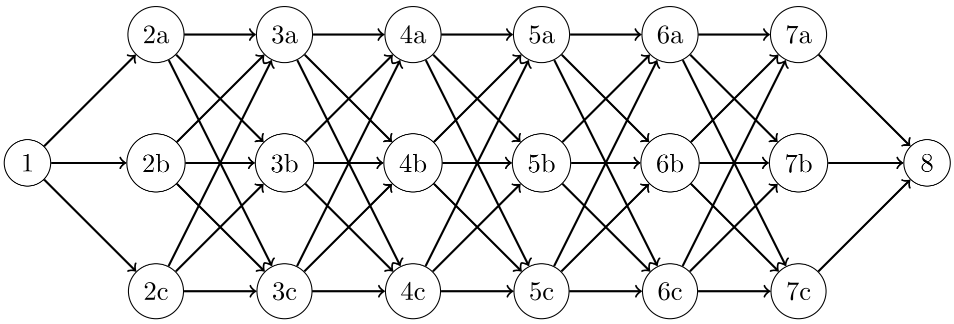
Suppose we have a DAG $G = (V, A)$ in which each arc $(i, j) \in A$ is associated with $Q^{i \to j} \in \mathcal{Q}$. A computation of interest is: given the last node $t \in V$ in a topological ordering π, calculate

$$\mu(s) := \sum_{(a_1 \ldots a_n) \in P(s,t)} \left(Q^{a_n^h \to a_n^t} * \cdots * Q^{a_1^h \to a_1^t} \right) \tag{17}$$

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

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

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

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

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

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

  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 (17) for all $s \in \rho(t)$

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

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

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 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 $\pi \in \Pi_G$

- $a^\omega \leftarrow$ **forward-rec**$(G, \omega, a_I, \pi \leftarrow ())$
- Return (a^ω, π).

forward-rec
Input: computation graph $G = (V, A)$, $i \in V$, input value a_I, topological ordering in construction π

- If $i \in V_I$ or a^i has already been calculated, just return a^i.
- Otherwise,
  - Calculate $a^j \leftarrow$ **forward-rec**(G, a_I, j, π) for each $j \in \mathbf{pa}(i)$.
  - Set $\pi \leftarrow \pi + (i)$ and return $x^i \leftarrow f^i\left(\left(a^j\right)_{j \in \mathbf{pa}(i)}\right)$.

H Example Functions in a Computation Graph

H.1 Common Functions

$$
\begin{aligned}
\textbf{add} &: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d & \textbf{add}(x, x') &:= x + x' \\
\textbf{cmult} &: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d & \textbf{cmult}(x, x') &:= x \odot x' \\
\textbf{concat} &: \mathbb{R}^d \times \mathbb{R}^{d'} \to \mathbb{R}^{d+d'} & \textbf{concat}(x, x') &:= x \oplus x' \\
\textbf{mult} &: \mathbb{R}^{d\times d''} \times \mathbb{R}^{d''\times d'} \to \mathbb{R}^{d\times d'} & \textbf{mult}(U, V) &:= UV \\
\textbf{pick} &: \mathbb{R}^d \times \{1 \dots d\} \to \mathbb{R} & \textbf{pick}(x, l) &:= x_l \\
\textbf{pnls} &: \mathbb{R}^d \times \mathbb{Z} \to \mathbb{R} & \textbf{pnls}(x, l) &:= \log\left(\sum_{j=1}^{d} \exp(x_j)\right) - x_l \\
\textbf{pow} &: \mathbb{R}^d \times \mathbb{Z} \to \mathbb{R}^d & \textbf{pow}_i(x, n) &:= x_i^n \\
\textbf{tanh} &: \mathbb{R}^d \to \mathbb{R}^d & \textbf{tanh}_i(x) &:= \tanh(x_i) \\
\textbf{logit} &: \mathbb{R}^d \to \mathbb{R}^d & \textbf{logit}_i(x) &:= \frac{1}{1 + \exp(-x_i)} \\
\textbf{sm} &: \mathbb{R}^d \to \mathbb{R}^d & \textbf{sm}_i(x) &:= \frac{\exp(x_i)}{\sum_{j=1}^{d} \exp(x_j)}
\end{aligned}
$$

H.2 Jacobians

Multi-argument functions.

$$
\begin{aligned}
&\textbf{add} & \frac{d(x+x')}{dx} &= I_{d\times d} & \frac{d(x+x')}{dx'} &= I_{d\times d} \\
&\textbf{cmult} & \frac{d(x\odot x')}{dx} &= \mathrm{diag}(x') & \frac{d(x\odot x')}{dx'} &= \mathrm{diag}(x) \\
&\textbf{concat} & \frac{d(x\oplus x')}{dx} &= \begin{bmatrix} I_{d\times d} \\ 0_{d'\times d} \end{bmatrix} & \frac{d(x\oplus x')}{dx'} &= \begin{bmatrix} 0_{d\times d'} \\ I_{d'\times d'} \end{bmatrix} \\
&\textbf{mult} & \frac{d[UV]_{i,j}}{dU_{k,l}} &= \begin{cases} V_{l,j} & \text{if } i = k \\ 0 & \text{else} \end{cases} & \frac{d[UV]_{i,j}}{dV_{k,l}} &= \begin{cases} U_{i,k} & \text{if } j = l \\ 0 & \text{else} \end{cases} \\
&\textbf{pick} & \frac{d\textbf{pick}(x,l)}{dx} &= e_l \\
&\textbf{pnls} & \frac{d\textbf{pnls}(x,l)}{dx_i} &= \begin{cases} \textbf{sm}_i(x) - 1 & \text{if } i = l \\ \textbf{sm}_i(x) & \text{else} \end{cases} \\
&\textbf{pow} & \frac{d\textbf{pow}_i(x,n)}{dx_j} &= \begin{cases} n \times x_i^{n-1} & \text{if } i = j \\ 0 & \text{else} \end{cases}
\end{aligned}
$$

Single-argument functions.

$$\textbf{tanh} \qquad \frac{d\,\textbf{tanh}_i(x)}{dx_j} = \begin{cases} 1 - \tanh(x_i)^2 & \text{if } i = j \\ 0 & \text{else} \end{cases}$$

$$\textbf{logit} \qquad \frac{d\,\textbf{logit}(x)_i}{dx_j} = \begin{cases} \textbf{logit}_i(x) \times (1 - \textbf{logit}_i(x)) & \text{if } i = j \\ 0 & \text{else} \end{cases}$$

$$\textbf{sm} \qquad \frac{d\,\textbf{sm}_i(x)}{dx_j} = \begin{cases} \textbf{sm}_i(x) \times (1 - \textbf{sm}_i(x)) & \text{if } i = j \\ -\textbf{sm}_i(x) \times \textbf{sm}_j(x) & \text{else} \end{cases}$$

I Practical Issues

I.1 Shape

Although we have followed the standard notation in vector calculus and defined the Jacobian of $f^j : \bigtimes_{t \in \mathbf{pa}(j)} \mathbb{R}^{d^t} \to \mathbb{R}^{d^j}$ with respect to x^i to be a $(d^j \times d^i)$ matrix so that $z^i \in \mathbb{R}^{1 \times d^i}$ in (2) is a *row* vector, in practice we want to make z^i a column vector to match the shape of $x^i \in \mathbb{R}^{d^i}$ (which we usually assume as a column vector). This is easily achieved by working with the transpose of (2). This means that we directly compute z^i as a column vector of length d^i given by summing over products of a $(d^i \times d^j)$ matrix and a column vector of length d^j,

$$z^i = \sum_{j \in \mathbf{ch}(i)} J^j \times z^j \tag{18}$$

where $J^j \in \mathbb{R}^{d^i \times d^j}$ is the transpose of the Jacobian, that is

$$J^j_{k,l} = \left. \frac{df^j_l(x^j_I)}{dx^i_k} \right|_{x^j_I = a^j_I}$$

I.2 Propagating a Linear Transformation of the Gradient

Consider any node with a local function f with output dimension d. For each of its parent variables $p \in \mathbb{R}^{d^p}$, let $g^p \in \mathbb{R}^{d^p}$ denote the gradient of p initialized to zero. Assuming that the gradient vector of the current node $g \in \mathbb{R}^d$ is complete, in light of (18) and the reverse topological traversal in backpropagation, the *only* calculation we need to perform is: for each parent variable p,

$$g^p \leftarrow g^p + J^p_f g$$

where $J^p_f \in \mathbb{R}^{d^p \times d}$ denotes the Jacobian of f with respect to p. Thus a central computational issue is to calculate the matrix-vector product $J^p_f g$ as efficiently as possible. Rather than explicitly calculating the matrix J^p_f and then calculating the product, we use the closed-form expressions given below (obtained by using Jacobians in Section H.2).

Multi-argument functions.

$$
\begin{array}{rll}
\textbf{add}\ (g \in \mathbb{R}^d) & J^{x}_{(x+x')} g = g & J^{x'}_{(x+x')} g = g \\
\textbf{cmult}\ (g \in \mathbb{R}^d) & J^{x}_{(x\odot x')} g = x' \odot g & J^{x'}_{(x\odot x')} g = x \odot g \\
\textbf{concat}\ (g \in \mathbb{R}^{d+d'}) & J^{x}_{(x\oplus x')} g = g_{1:d} & J^{x}_{(x\oplus x')} g = g_{d+1:d'} \\
\textbf{mult}\ (g \in \mathbb{R}^{d\times d'}) & J^{U}_{(UV)} g = gV^\top & J^{V}_{(UV)} g = U^\top g \\
\textbf{pick}\ (g \in \mathbb{R}) & J^{x}_{\textbf{pick}(x,l)} g = g e_l & \\
\textbf{pnls}\ (g \in \mathbb{R}) & J^{x}_{\textbf{pnls}(x,l)} g = g(\textbf{sm}(x) - e_l) & \\
\textbf{pow}\ (g \in \mathbb{R}^d) & J^{x}_{\textbf{pow}(x,n)} g = n\textbf{pow}(x, n-1) \odot g &
\end{array}
$$

Single-argument functions.

$$
\textbf{sm}\ (g \in \mathbb{R}^d) \qquad J^{x}_{\textbf{sm}(x)} = \textbf{sm}(x) \odot g - \left(\sum_{i=1}^{d} [\textbf{sm}(x) \odot g]_i \right) \textbf{sm}(x)
$$