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

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

For convenience, we will define

$$x_I := (x^i)_{i \in V_I}$$
$$a_I := (a^i)_{i \in V_I}$$
$$x^i_I := (x^j)_{j \in \text{pa}(i)}$$
$$a^i_I := (a^j)_{j \in \text{pa}(i)}$$

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 $x_I$,

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

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:

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

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 $i \in V$.

*Code: https://github.com/karlstratos/simplenet
3 Backpropagation

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

$$z^i := \frac{dL^\omega(x_I)}{dx^i} \bigg|_{x_I = a_I} \in \mathbb{R}^{1 \times d^i}$$ (1)

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

$$z^i = \sum_{j \in \text{ch}(i)} \frac{dL^\omega(x_I)}{dx^j} \bigg|_{x_I = a_I} \cdot \frac{dx^j}{dx^i} \bigg|_{x_I = a_I} \in \mathbb{R}^{1 \times d^i} \, \text{(2)}$$

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 $i \in \text{pa}(j)$, this can be analytically computed. For instance, if $f^j = \text{cmult}$ where $\text{cmult} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ is defined as $\text{cmult}(x, x') := x \odot x'$ (Appendix H), then the Jacobian of $\text{cmult}$, evaluated at $(x, x') = (a, a')$, with respect to $x$ is

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

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

$$\frac{dL^\omega(x_I)}{dx^\omega} \bigg|_{x_I = a_I} = \frac{dx^\omega}{dx^1} \bigg|_{x_I^1 = a_I^1} = 1$$

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
    $$z^i := \sum_{j \in \text{ch}(i)} z^j \frac{dx^i}{dx^j} \bigg|_{x_I^j = a_I^j}$$

As observed in the note by Michael Collins, another way to view the algorithm is a sum-product algorithm on a DAG (Appendix F.2) since

$$\frac{dL^\omega(x_I)}{dx^i} \bigg|_{x_I^1 = a_I^1} = \sum_{(i_1 \ldots i_n) \in P(i, \omega)} \frac{dx^\omega}{dx^{i_n-1}} \bigg|_{x_I^{i_n-1} = a_I^{i_n-1}} \cdot \cdots \cdot \frac{dx^2}{dx^{i_1}} \bigg|_{x_I^{i_1} = a_I^{i_1}}$$

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 \(S^n := \{v \in \mathbb{R}^n : \|v\|_2 = 1\}\). 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 \(\sigma(x) := \frac{1}{1 + \exp(-x)} - 1\). 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 \(f : \mathbb{R} \to \mathbb{R}\) 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

\[
\lim_{x \to a} f(x) = L
\]  

(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

\[
\lim_{x \to a} \frac{f(x) - (f(a) + f'(a)(x - a))}{x - a} = 0
\]  

(4)

This definition is equivalent to

\[
f'(a) := \lim_{\epsilon \to 0} \frac{f(a + \epsilon) - f(a)}{\epsilon}
\]  

(5)

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

B.3 Chain Rule

We write

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

The central tool for this problem is the chain rule
\[
\left. \frac{dg(f(x))}{dx} \right|_{x=a} = \frac{dg(y)}{dy} \bigg|_{y=f(a)} \times \frac{df(x)}{dx} \bigg|_{x=a}
\]
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)
\[
g(y) \approx g(b) + g'(b)(y - b) \quad \forall y, b \in \mathbb{R}
\]
\[
f(x) \approx f(a) + f'(a)(x - a) \quad \forall x, a \in \mathbb{R}
\]
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 \( x \).

\section{Scalar-Valued Function of a Vector Variable}

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

\subsection{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.

\subsection{Directional/Partial Derivative}

The directional derivative of \( f \) at \( a \) in the direction of \( v \in S^n \) is
\[
D_v f(a) := f'_v(0) = \lim_{\epsilon \to 0} \frac{f(a + \epsilon v) - f(a)}{\epsilon}
\]
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 \( e_i \):
\[
\frac{\partial f(a)}{\partial x_i} := D_{e_i} f(a) = \lim_{\epsilon \to 0} \frac{f(a_1, \ldots, a_i + \epsilon, \ldots, a_n) - f(a_1, \ldots, a_n)}{\epsilon}
\]

\subsection{Gradient}

The gradient of \( f \) at \( a \) is a unique vector \( \nabla f(a) \in \mathbb{R}^n \) such that
\[
\lim_{x \to a} \frac{f(x) - (f(a) + \nabla f(a)^T(x - a))}{||x - a||} = 0
\]
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

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

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

\[
\nabla f(a) = \begin{pmatrix}
\frac{\partial f(a)}{\partial x_1} \\
\vdots \\
\frac{\partial f(a)}{\partial x_n}
\end{pmatrix}
\]

(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 \( i \in \{1 \ldots n\} \) 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 \( f_1 \ldots f_m : \mathbb{R}^n \to \mathbb{R} \). 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 \( T^f_a \in \mathbb{R}^{m \times n} \) such that

\[
\lim_{x \to a} \frac{||f(x) - (f(a) + T^f_a (x - a))||}{||x - a||} = 0
\]

(11)

This is a natural generalization of the gradient (8) (equivalent when \( m = 1 \)): the linear function \( f(a) + T^f_a (x - a) \) is a linear approximation of \( f(x) \) around \( a \). We say \( f \) is **differentiable at** \( a \) if \( T^f_a \) exists.
D.2 Total Derivative as Jacobian

It is easy to see that when \( f_1 \ldots f_m \) are differentiable, we have

\[
T^f_a = \begin{bmatrix}
\nabla f_1(a)^T \\
\vdots \\
\nabla f_m(a)^T
\end{bmatrix} =: J_f(a) \quad (12)
\]

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

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

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 \( f : \mathbb{R}^n \rightarrow \mathbb{R}^m \) 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 (14):

\[
\left. \frac{dg(f(x))}{dx} \right|_{x=a} = \left. \frac{dg(y)}{dy} \right|_{y=f(a)} \left. \frac{df(x)}{dx} \right|_{x=a} \quad (14)
\]

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

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

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 \( g(y_1 \ldots y_m) \).
**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

\[ f : \mathbb{R}^{n_1 \times \cdots \times n_N} \rightarrow \mathbb{R}^{m_1 \times \cdots \times m_M} \]

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

\[ i \in \{1 \cdots (n_1 \cdots n_N)\} \]

are in one-to-one correspondence with tuples

\[ (i_1, \ldots, i_M) \in \{1 \cdots n_1\} \times \cdots \times \{1 \cdots n_M\} \]

Let \( \text{ind}(i_1, \ldots, i_M) \) denote vector index corresponding to the tensor index tuple \((i_1, \ldots, i_M)\). 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

\[
\begin{bmatrix}
\frac{df(X)}{dX} \\
X = A
\end{bmatrix}_{\text{ind}(i_1, \ldots, i_M), \text{ind}(j_1, \ldots, j_N)} = \frac{d f_{i_1 \ldots i_M}(X)}{d X_{j_1 \ldots j_N}} \bigg|_{X = A}
\]

**(16)**

**Chain rule.** Suppose we introduce

\[ g : \mathbb{R}^{m_1 \times \cdots \times m_M} \rightarrow \mathbb{R}^{d_1 \times \cdots \times d_D} \]

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

\[
\frac{dg(f(X))}{dX} \bigg|_{X = A} = \frac{dg(y)}{dy} \bigg|_{y = f(A)} \frac{df(X)}{dX} \bigg|_{X = A}
\]

Equivalently, the chain rule states that the total derivative of \( g \) on \( f \) 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 \((i_1, \ldots, i_D, j_1, \ldots, j_N)\)-th element is

\[
\frac{dg_{i_1 \ldots i_D}(f(X))}{dX_{j_1 \ldots j_N}} \bigg|_{X = A} = \sum_{k_1 = 1}^{m_1} \cdots \sum_{k_M = 1}^{m_M} \frac{dg_{i_1 \ldots i_D}(B)}{dB_{k_1 \ldots k_M}} \bigg|_{B = f(A)} \times \frac{df_{k_1 \ldots k_M}(X)}{dX_{j_1 \ldots j_N}} \bigg|_{X = A}
\]
F Directed Acyclic Graph (DAG)

F.1 Terminology

A **directed graph** is a pair $G = (V, A)$ where $V = \{1, \ldots, |V|\}$ is a set of nodes and $A \subseteq V \times V$ 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 $\text{pa}(i) := \{j \in V : (j, i) \in A\}$ and the set of its children by $\text{ch}(i) := \{j \in V : (i, j) \in A\}$. We say $i \in V$ is an **input node** if $\text{pa}(i) = \emptyset$. 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 $\text{ch}(i) = \emptyset$.

The set of **paths** from $i \in V$ to $j \in V$ where $i \neq j$ is

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

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):

```
1 --2 --3 --4 --5
  \  \\
   \  \\
   \  \\
   \  \\
   \  \\
    \  \\
     \  \\
     \  \\
     \  \\
     \  \\
     6
```

- $V = \{1, 2, 3, 4, 5, 6\}$
- $A = \{(1, 3), (1, 5), (2, 4), (3, 4), (4, 6), (5, 6)\}$
- $\text{pa}(4) = \{2, 3\}$
- $\text{ch}(1) = \{3, 5\}$
- $\Pi_G = \{(1, 2, 3, 4, 5, 6), (2, 1, 3, 4, 5, 6)\}$
- $V_I = \{1, 2\}$
- $V_N = \{3, 4, 5, 6\}$
- $P(1, 6) = \{((1, 3), (3, 4), (4, 6)), ((1, 5), (5, 6))\}$
- $P(2, 6) = \{((2, 4), (4, 6))\}$
- $P(2, 5) = \emptyset$
- $\rho(6) = \{1, 2, 3, 4, 5\}$
- $\rho(5) = \{1\}$
- $\rho(4) = \{1, 2, 3\}$

F.2 Sum-Product Algorithm on DAGs

Let $Q$ be any set equipped with associative binary operations $+$ and $\ast$. We assume that the multiplicative operation $\ast$ is distributive over $+$. We assume that the additive operation $+$ is commutative but the multiplicative operation $\ast$ may not be. For
instance, \( Q \) 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 \( Q^{i \to j} \in Q \). A computation of interest is: given the last node \( t \in V \) in a topological ordering \( \pi \), calculate

\[
\mu(s) := \sum_{(a_1, \ldots, a_n) \in P(s,t)} \left( Q^{a_n \to t} \ast \ldots \ast Q^{a_1 \to s} \right)
\]

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

\[
\mu(1) = (Q^{4 \to 6} \ast Q^{3 \to 4} \ast Q^{1 \to 3}) + (Q^{5 \to 6} \ast Q^{1 \to 5})
\]

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

![DAG Diagram](image)

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

- If \( s \in \text{pa}(t) \):
  \[
  \mu(s) = Q^{s \to t}
  \]

- If \( s \not\in \text{pa}(t) \):
  
  \[
  \mu(s) = \sum_{(a_1, \ldots, a_n) \in P(s,t)} \left( Q^{a_n \to t} \ast \ldots \ast Q^{a_1 \to s} \right)
  \]
  
  \[
  = \sum_{i \in \text{ch}(s)} \left( \sum_{(a_2, \ldots, a_n) \in P(i,t)} \left( Q^{a_n \to t} \ast \ldots \ast Q^{a_2 \to s} \ast Q^{a_1 \to s} \right) \right)
  \]
  
  \[
  = \sum_{i \in \text{ch}(s)} \mu(i) \ast Q^{s \to i}
  \]

  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 $i = |V| - 1 \ldots 1$, set $s = \pi(i)$ and compute
  
  $$\mu(s) = \begin{cases} 
  Q^{s \rightarrow i} & \text{if } s \in \text{pa}(t) \\
  \sum_{j \in \text{ch}(s)} \mu(j) * Q^{s \rightarrow j} & \text{else}
  \end{cases}$$

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: $L^{\omega}(x_I) := x^{\omega}$ 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 $i \in V_I$ or $a^i$ has already been calculated, just return $a^i$.
- Otherwise,
  - Calculate $a^i \leftarrow \text{forward-rec}(G,a_I,j,\pi)$ for each $j \in \text{pa}(i)$.
  - Set $\pi \leftarrow \pi + (i)$ and return $x^i \leftarrow f^i \left( (a^i)_{j \in \text{pa}(i)} \right)$.
H Example Functions in a Computation Graph

H.1 Common Functions

\[ \text{add} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d \]
\[ \text{cmult} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d \]
\[ \text{concat} : \mathbb{R}^d \times \mathbb{R}^{d'} \rightarrow \mathbb{R}^{d+d'} \]
\[ \text{mult} : \mathbb{R}^{d \times d''} \times \mathbb{R}^{d' \times d''} \rightarrow \mathbb{R}^{d \times d'} \]
\[ \text{pick} : \mathbb{R}^d \times \{1, \ldots, d\} \rightarrow \mathbb{R} \]
\[ \text{pnl} : \mathbb{R}^{d \times \mathbb{Z}} \rightarrow \mathbb{R} \]
\[ \text{pow} : \mathbb{R}^d \times \mathbb{Z} \rightarrow \mathbb{R}^d \]
\[ \text{tanh} : \mathbb{R}^d \rightarrow \mathbb{R}^d \]
\[ \text{logit} : \mathbb{R}^d \rightarrow \mathbb{R}^d \]
\[ \text{sm} : \mathbb{R}^d \rightarrow \mathbb{R}^d \]

H.2 Jacobians

Multi-argument functions.

\[ \text{add} \quad \frac{d(x + x')}{dx} = I_{d \times d} \quad \frac{d(x + x')}{dx'} = I_{d \times d} \]
\[ \text{cmult} \quad \frac{d(x \odot x')}{dx} = \text{diag}(x') \quad \frac{d(x \odot x')}{dx'} = \text{diag}(x) \]
\[ \text{concat} \quad \frac{d(x \oplus x')}{dx} = \begin{bmatrix} I_{d \times d} \\ 0_{d' \times d} \end{bmatrix} \quad \frac{d(x \oplus x')}{dx'} = \begin{bmatrix} 0_{d \times d'} \\ I_{d' \times d'} \end{bmatrix} \]
\[ \text{mult} \quad \frac{d[UV]_{i,j}}{dU_{k,l}} = \begin{cases} V_{i,j} & \text{if } i = k \\ 0 & \text{else} \end{cases} \quad \frac{d[UV]_{i,j}}{dV_{k,l}} = \begin{cases} U_{i,k} & \text{if } j = l \\ 0 & \text{else} \end{cases} \]
\[ \text{pick} \quad \frac{dpick(x, l)}{dx} = \delta_l \]
\[ \text{pnl} \quad \frac{dpnls(x, l)}{dx_i} = \begin{cases} \text{sm}_i(x) - 1 & \text{if } i = l \\ \text{sm}_i(x) & \text{else} \end{cases} \]
\[ \text{pow} \quad \frac{dpow(x, n)}{dx_j} = \begin{cases} n \times x_i^{n-1} & \text{if } i = j \\ 0 & \text{else} \end{cases} \]
Single-argument functions.

\[
\begin{align*}
\frac{dtanh(x)}{dx} &= \begin{cases} 
    1 - \tanh(x_i)^2 & \text{if } i = j \\
    0 & \text{else}
\end{cases} \\
\frac{d\logit(x)}{dx} &= \begin{cases} 
    \logit_i(x) \times (1 - \logit_i(x)) & \text{if } i = j \\
    0 & \text{else}
\end{cases} \\
\frac{dsm_i(x)}{dx} &= \begin{cases} 
    sm_i(x) \times (1 - sm_i(x)) & \text{if } i = j \\
    -sm_i(x) \times sm_j(x) & \text{else}
\end{cases}
\end{align*}
\]

I. Practical Issues

I.1 Shape

Although we have followed the standard notation in vector calculus and defined the Jacobian of \( f_j : x_{i \in pa(j)} \rightarrow R^d \) with respect to \( x^i \) to be a \((d_j \times d^i)\) matrix so that \( z^i \in R^{1 \times d^i} \) in (2) is a row vector, in practice we want to make \( z^i \) a column vector to match the shape of \( x^i \in R^{d^i} \) (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 \( d^j \),

\[
z^i = \sum_{j \in ch(i)} J^j \times z^j 
\]

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

\[
J^j_{k,l} = \frac{df_j^l(x_j^i)}{dx_k^j} \bigg|_{x_j^i = a_j^i} 
\]

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 \( p \in R^{d^p} \), let \( g^p \in R^{d^p} \) denote the gradient of \( p \) initialized to zero. Assuming that the gradient vector of the current node \( g \in R^d \) 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 \( p \),

\[
g^p \leftarrow g^p + J_f^p g
\]

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

\[
\text{add} \ (g \in \mathbb{R}^d) \quad J^x_{(x+x')} g = g \\
\text{cmult} \ (g \in \mathbb{R}^d) \quad J^x_{(x\odot x')} g = x' \odot g \\
\text{concat} \ (g \in \mathbb{R}^{d+d'}) \quad J^x_{(x\oplus x')} g = g_{1:d} \\
\text{mult} \ (g \in \mathbb{R}^{d\times d'}) \quad J^x_{(UV)} g = gV^\top \\
\text{pick} \ (g \in \mathbb{R}) \quad J^x_{\text{pick}(x,l)} g = ge_l \\
\text{pnls} \ (g \in \mathbb{R}) \quad J^x_{\text{pnls}(x,l)} g = g(\text{sm}(x) - e_l) \\
\text{pow} \ (g \in \mathbb{R}^d) \quad J^x_{\text{pow}(x,n)} g = n\text{pow}(x, n-1) \odot g
\]

Single-argument functions.

\[
\text{sm} \ (g \in \mathbb{R}^d) \quad J^x_{\text{sm}(x)} = \text{sm}(x) \odot g - \left( \sum_{i=1}^d [\text{sm}(x) \odot g]_i \right) \text{sm}(x)
\]