Chapter 1

Computational Graphs, and Backpropagation

(Course notes for NLP by Michael Collins, Columbia University)

1.1 Introduction

We now describe the *backpropagation* algorithm for calculation of derivatives in neural networks. We have seen in the previous note how these derivatives can be used in conjunction with stochastic gradient descent to train the parameters of a neural network.

We will also introduce *computational graphs*, a formalism that will allow us to specify a wide range of neural network models. We will describe how *automatic differentiation* can be implemented on computational graphs, allowing backpropagation to be derived "for free" once a network has been expressed as a computational graph.

1.2 Computational Graphs

Computational graphs are a powerful formalism that have been extremely fruitful in deriving algorithms and software packages for neural networks and other models in machine learning. The basic idea in a computational graph is to express some model—for example a feedforward neural network—as a directed graph expressing a sequence of computational steps. Each step in the sequence corresponds to a vertex in the computational graph; each step corresponds to a simple operation that takes some inputs and produces some output as a function of its inputs. Directed edges in the graph are used to specify the inputs to each vertex.

Two key strengths of computational graphs are as follows. The first is that they allow simple functions to be combined to form quite complex models: a wide range of neural network models can be created by defining computational graphs consisting of simple, primitive operations. The second strength is that they enable *automatic differentiation*. We saw in the previous note on neural networks that gradient-based learning algorithms such as stochastic gradient descent can be used to train neural networks, providing that the gradient of the loss function with respect to the parameters can be calculated efficiently. Automatic differentiation is a technique for calculating derivatives in computational graphs: once the graph has been defined using underlying primitive operations, derivatives are calculated automatically based on "local" derivatives of these operations.

For convenience, figure 1.1 contains a glossary of important terms connected with computational graphs. Over the remainder of this note we will explain all terms in the glossary.

1.2.1 Notation

Throughout this note we use lower-case letters such as f, d or u to refer to scalars or vectors. For a vector u we use u_i to refer to the i'th component of u. We use upper-case letters such as P to refer to matrices. We will frequently use superscripts on scalar, vector, or matrix variables: for example u^2 is a scalar or vector (it is not the case that $u^2 = u \times u$). In cases where we do need to refer to exponentiation, we will use parantheses around the term being exponentiated: for example $(u^2)^3$ is equal to the value u^2 raised to the third power.

1.2.2 Computational Graphs: A Formal Definition

We define computational graphs as follows:

Definition 1 (Computational Graphs) *A computational graph is a 6-tuple*

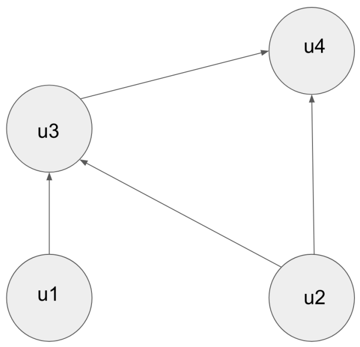
$$\langle n, l, E, u^1 \ldots u^n, d^1 \ldots d^n, f^{l+1} \ldots f^n \rangle$$

where:

- *n is an integer specifying the number of vertices in the graph.*
- *l is an integer such that $1 \leq l < n$ that specifies the number of leaves in the graph.*

- n: The number of vertices in the computational graph.
- l: The number of leaves in the computational graph. We have $1 \leq l < n$.
- E: The set of edges in the computational graph. For each $(j, i) \in E$ we have $j < i$ (the graph is topologically ordered), $j \in \{1 \ldots (n-1)\}$, and $i \in \{(l+1) \ldots n\}$.
- u^i for $i \in \{1 \ldots n\}$ is the variable associated with vertex i in the graph.
- d^i for $i \in \{1 \ldots n\}$ is the dimensionality for each variable, that is, $u^i \in \mathbb{R}^{d^i}$.
- f^i for $i \in \{(l+1) \ldots n\}$ is the local function for vertex i in the graph
- α^i for $i \in \{(l+1) \ldots n\}$ is defined as $\alpha^i = \langle u^j | (j, i) \in E \rangle$, i.e., α^i contains all input values for vertex i.
- **The forward algorithm:** Given leaf values $u^1 \ldots u^l$, calculates $u^{l+1} \ldots u^n$ through: for $i = (l+1) \ldots n$, $u^i = f^i(\alpha^i)$ where $\alpha^i = \langle u^j | (j, i) \in E \rangle$.
- h^i for $i \in \{(l+1) \ldots n\}$ is the global function for vertex i in the graph. The forward algorithm calculates $u^i = h^i(u^1 \ldots u^l)$ for each $i \in \{(l+1) \ldots n\}$.
- $J^{j \to i}$ for $(j, i) \in E$ is the local Jacobian function for edge (j, i) in the graph. We define
$$J^{j \to i}(\alpha^i) = \frac{\partial f^i(\alpha^i)}{\partial u^j}$$
- $I(d)$: the identiy matrix of dimension $d \times d$.
- **The backpropagation algorithm:** Given leaf values $u^l \ldots u^l$, and values for $u^{l+1} \ldots u^n$ calculated using the forward algorithm, sets $P^n = I(d^n)$, and for $j = (n-1) \ldots 1$:
$$P^j = \sum_i P^i J^{j \to i}(\alpha^i)$$
The output from the algorithm is P^j for $j \in \{1 \ldots l\}$, with
$$P^j = \frac{\partial h^n(u^1 \ldots u^l)}{\partial u^j}$$
i.e., P^j is a Jacobian summarizing the partial derivatives of the output h^n with respect to the input u^j.

Figure 1.1: A glossary of important terms in computational graphs.



$$\begin{aligned}
d^i &= 1 \quad \text{for } i = 1 \ldots 4 \\
f^3(u^1, u^2) &= u^1 + u^2 \\
f^4(u^2, u^3) &= u^2 \times u^3
\end{aligned}$$

Figure 1.2: A computational graph with $n = 4$, $l = 2$, $E = \{(1,3),(2,3),(2,4),(3,4)\}$. We show the graph and the values for $d^1 \ldots d^n$ and $f^1 \ldots f^n$.

- *E is a set of directed edges in the graph. Each edge is an ordered pair (j, i) where $j \in \{1 \ldots (n-1)\}$, $i \in \{(l+1) \ldots n\}$, and $j < i$.*

- *Each u^i for $i = 1 \ldots n$ is a* **variable** *associated with vertex i in the graph. u^i is a variable in $\mathbb{R}^{d^i}$, hence d^i for $i = 1 \ldots n$ is an integer specifying the dimensionality of the variable associated with variable i.*

- *We take u^n to be the* **output variable** *in the graph. Often we have $d^n = 1$, so that the output variable is a scalar; however we will consider the general case where d^n is any integer greater than 0.*

- *Each f^i for $i = (l+1) \ldots n$ is a* **local function** *associated with vertex i in the graph. If we define*

  $$\alpha^i = \langle u^j | (j,i) \in E \rangle$$

  hence α^i is a vector formed by concatenating the vectors u^j such that $(j,i) \in E$, and in addition we define

  $$\bar{d}^i = \sum_{j:(j,i) \in E} d^j$$

  then we have dimensions $\alpha^i \in \mathbb{R}^{\bar{d}^i}$, $f^i : \mathbb{R}^{\bar{d}^i} \rightarrow \mathbb{R}^{d^i}$, and $f^i(\alpha^i) \in \mathbb{R}^{d^i}$.

$\square$

Note that the condition that each edge (j,i) has $j < i$ implies that the graph is *topologically ordered*. The condition that each edge (j,i) has $j \in \{1 \ldots (n-1)\}$ implies that the output variable u^n has no outgoing arcs. The condition that each edge (j, i) has $i \in \{(l+1) \dots n\}$ implies that each leaf node $j \in \{1 \dots l\}$ has zero incoming arcs.

Example 1 (An Example Computational Graph) *Consider the following example. We have $n = 4$ and $l = 2$. We define the set of directed edges to be $E = \{(1,3),(2,3),(2,4),(3,4)\}$. We define $d^i = 1$ for $i = 1 \dots 4$: it follows that each variable u^i is a one-dimensional vector (i.e., a scalar). We define the local functions as follows:*

$$\begin{aligned} f^3(u^1, u^2) &= u^1 + u^2 \\ f^4(u^2, u^3) &= u^2 \times u^3 \end{aligned}$$

See figure 1.3 for an illustration of this computational graph.

1.2.3 The Forward Algorithm in Computational Graphs

Figure 1.3 shows the forward algorithm in computational graphs. The algorithm takes as input a computational graph together with values for the leaf variables $u^1 \dots u^l$. It returns a value for u^n as its output. For each variable u^i for $i = (l+1) \dots n$, the value is calculated as

$$u^i = f^i(\alpha^i)$$

where

$$\alpha^i = \langle u^j | (j, i) \in E \rangle$$

Note that because the graph is topologically ordered (i.e., for any $(j, i) \in E$, we have $j < i$), the values for each u^j such that $(j, i) \in E$ are calculated before u^i is calculated.

Example 2 (An Example of the Forward Algorithm.) *As an example of the forward algorithm, consider applying it to the computational graph in figure 1.2, with input leaf values $u^1 = 2$ and $u^2 = 3$. Then the algorithm proceeds with the following steps:*

$$u^3 = f^3(u^1, u^2) = u^1 + u^2 = 5$$

$$u^4 = f^4(u^2, u^3) = u^2 \times u^3 = 15$$

$$\textit{Output} \ = u^4 = 15$$

Input: A computational graph $G = \langle n, l, E, d^1 \ldots d^n, u^1 \ldots u^n, f^1 \ldots f^n \rangle$. Values for the leaf variables $u^1 \ldots u^l$.
Algorithm:
For $i = (l+1) \ldots n$, compute

$$u^i = f^i(\alpha^i)$$

where

$$\alpha^i = \langle u^j | (j,i) \in E \rangle$$

Output: u^n

Figure 1.3: The forward algorithm. The algorithm takes as inputs values for the leaf variables $u^1 \ldots u^l$, and returns u^n as the output value.

1.2.4 The Global Functions Defined by a Computational Graph

It will be useful to explicitly define the functions that map a set of leaf values $u^1 \ldots u^l$ to non-leaf values u^i for $i \in \{(l+1) \ldots n\}$, as follows:

Definition 2 (Global Functions) *Given a computational graph, the global functions $h^i(u^1 \ldots u^l)$ for $i = 1 \ldots n$ are defined as follows:*

If $i \in \{1 \ldots l\}$,

$$h^i(u^1 \ldots u^l) = u^i$$

else if $i \in (l+1) \ldots n$,

$$h^i(u^1 \ldots u^l) = f^i(\alpha^i)$$

where

$$\alpha^i = \langle h^j(u^1 \ldots u^l) | (j,i) \in E \rangle$$

Note that this is a recursive definition, with the base cases $h^i(u^1 \ldots u^l) = u^i$ for $i \in \{1 \ldots l\}$, and recursive definitions for $i \in \{(l+1) \ldots n\}$.

It can be verified that for $i \in \{(l+1) \ldots n\}$, the forward algorithm calculates values

$$u^i = h^i(u^1 \ldots u^l)$$

for $i = (l+1) \ldots n$. In particular, the output from the algorithm is

$$u^n = h^n(u^1 \ldots u^) l$$

1.2.5 A Single-Layer Feedforward Network as a Computational Graph

We now describe how a feedforward network with a single hidden layer can be specified as a computational graph. The network defines a model of the following form:

$$p(y^i|x^i; V, \gamma, W, b) = \frac{\exp\{V_{y^i} \cdot g(Wx^i + b) + \gamma_{y^i}\}}{\sum_{y'} \exp\{V_{y'} \cdot g(Wx^i + b) + \gamma_{y'}\}}$$

Here $x^i \in \mathbb{R}^d$ is the input to the network, and $y^i \in \{1 \ldots K\}$ is the output label, where K is an integer specifying the number of output labels. If the integer m specifies the number of neurons in the network, then $W \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$ are the parameters of the hidden layer, and $g : \mathbb{R}^m \to \mathbb{R}^m$ is the transfer function. For each label y, $V_y \in \mathbb{R}^m$ is a parameter vector for the label y, and γ_y is a bias value for the label y.

We will also find it convenient to use $V \in \mathbb{R}^{K \times m}$ to refer to a matrix with rows $V_1, V_2, \ldots V_K$, and $\gamma \in \mathbb{R}^K$ to be a vector of bias values.

Given this model, we can define the following function of the inputs (x^i, y^i) together with the parameters W, b, V, γ:

$$L(x^i, y^i, W, b, V, \gamma) = -\log p(y^i|x^i; V, \gamma, W, b)$$

We will now describe a computational graph that has $x^i, y^i, W, b, V, \gamma$ as its leaf variables, and has $L(x^i, y^i, W, b, V, \gamma)$ as the value of its output.

Figure 1.4 shows the computational graph. We make a couple of remarks:

- For ease of comprehension we have used variable names $W, x^i, b, y^i, V, \gamma, z, h, l, q$ and o rather than names $u^1, u^2, u^3, \ldots u^n$. However this is a minor detail: we could replace W with u^1, x^i with u^2, and so on.

- Two of the variables in the graph, W and V, are matrices rather than vectors, with dimensionalities $(m \times d)$ and $(K \times m)$ respectively. In a slight abuse of notation we will allow variables in computational graphs that are matrices in addition to vectors; we can effectively treat a matrix in $\mathbb{R}^{m \times d}$ as a vector with $(m \times d)$ elements, with the vector being indexed by pairs (m', d') where $1 \leq m' \leq m$ and $1 \leq d' \leq d$.

It can be seen that the values of the non-leaf variables are calculated as follows:

- $z = Wx^i + b$. This is the usual way of computing $z \in \mathbb{R}^m$ as the input to the m neurons in the single-layer model.

- $h = g(z)$. This computes the output vector h from the neurons.

- $l = Vh + \gamma$. If we define V_y for $y \in \{1 \ldots K\}$ to be the y'th row of the matrix V, then we have
$$l_y = V_y \cdot h + \gamma_y$$
hence l_y is the "logit" (unnormalized score) for label y under the model.

- $q = \text{LOG-SOFTMAX}(l)$. Here we define $\text{LOG-SOFTMAX}(l)$ such that
$$q_y = \log \frac{\exp\{l_y\}}{\sum_{y'} \exp\{l_{y'}\}} = l_y - \log \sum_{y'} \exp\{l_{y'}\}$$
Hence q_y is the log-probability for label y under the model.

- $o = -q_{y^i}$. The output from the network is the value of variable o, which is defined here to be the negative log probability of the label y^i.

The next question is how to compute the derivatives of
$$o = L(x^i, y^i, W, b, V, \gamma)$$
Specifically, we would like to compute partial derivatives
$$\frac{\partial o}{\partial W_{i,j}} \quad \text{for all } i, j$$
$$\frac{\partial o}{\partial b_i} \quad \text{for all } i$$
$$\frac{\partial o}{\partial V_{i,j}} \quad \text{for all } i, j$$
$$\frac{\partial o}{\partial \gamma_i} \quad \text{for all } i$$
Next we will describe how the backpropagation algorithm can be used to calculate partial derivatives of the output with respect to each of the leaf variables in a computational graph.

1.3 Partial Derivatives, and the Chain Rule

In the next section we will derive the backpropagation algorithm for computation of partial derivatives in computational graphs.

In this section we give important background: first describing notation and definitions for partial derivatives; then describing a first version of the chain rule, for scalar variables; then describing Jacobians for functions that have vector inputs and outputs; and finally giving a second version of the chain rule that makes use of Jacobians.

Number of variables: $n = 11$
Number of leaf variables: $l = 6$
Leaf variables:

$$\begin{aligned}
u^1 &= W \in \mathbb{R}^{m \times d} \\
u^2 &= b \in \mathbb{R}^{m} \\
u^3 &= x^i \in \mathbb{R}^{d} \\
u^4 &= y^i \in \{1 \ldots K\} \\
u^5 &= V \in \mathbb{R}^{K \times m} \\
u^6 &= \gamma \in \mathbb{R}^{K}
\end{aligned}$$

Non-leaf variables, and local functions $f^7 \ldots f^{11}$:

$$\begin{aligned}
u^7 &= z \in \mathbb{R}^m = Wx^i + b \\
u^8 &= h \in \mathbb{R}^m = g(z) \\
u^9 &= l \in \mathbb{R}^K = Vh + \gamma \\
u^{10} &= q \in \mathbb{R}^K = \text{LOG-SOFTMAX}(l) \\
u^{11} &= o \in \mathbb{R} = -q_{y^i}
\end{aligned}$$

Edges: $E = \{(1,7),(2,7),(3,7),(7,8),(5,9),(6,9),(8,9),(9,10),(4,11),(10,11)\}$
A figure showing the graph:

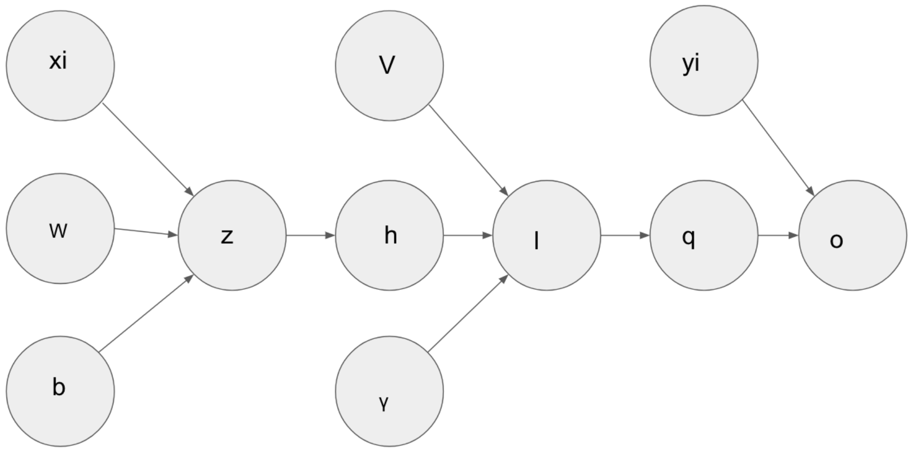


Figure 1.4: A computational graph for a single layer feedforward network that maps leaf values $x^i, y^i, W, b, V, \gamma$ to an output value $o = -\log p(y^i|x^i; W, b, V, \gamma)$.

1.3.1 Partial Derivatives

Given a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, and an equation

$$y = f(z^1, z^2, \ldots, z^n)$$

where y and z^i for $i = 1 \ldots n$ are scalars, we will write the partial derivative of the function f with respect to the variable z^i as

$$\frac{\partial f(z^1 \ldots z^n)}{\partial z^i}$$

We use the conventional definition of a partial derivative: this is the derivative of f with respect to parameter z^i, holding the other arguments to f (i.e., z^j for $j \neq i$) fixed. Or more precisely,

$$\frac{\partial f(z^1 \ldots z^n)}{\partial z^i} = \lim_{h \rightarrow 0} \frac{f(z^1 \ldots, z^i + h, \ldots z^n) - f(z^1 \ldots z^n)}{h}$$

We will also use the following notation to refer to the partial derivative:

$$\left.\frac{\partial y}{\partial z^i}\right|^f_{z^1 \ldots z^n}$$

which can be read as "the partial derivative of y with respect to z^i, under function f, at values $z^1 \ldots z^n$."

We are careful here that the notation explicitly identifies the function f that underlies the partial derivative. This is unconventional, but will help to avoid confusion. In deriving the backpropagation algorithm we will see that there are many different functions in the context, and it will help us to keep track of exactly which function we are talking about at each step.

We will sometimes drop f or $z^1 \ldots z^n$ when these values are clear from the context.

1.3.2 A First Version of the Chain Rule

Again consider a function

$$y = f(z^1, z^2, \ldots z^n)$$

where y and $z^1 \ldots z^n$ are scalars. Assume in addition that for $i = 1 \ldots n$

$$z^i = g^i(x^1 \ldots x^{n'})$$

where $x^1 \ldots x^{n'}$ are also scalars, and each g^i is a function $g^i : \mathbb{R} \rightarrow \mathbb{R}$. Define h to be the composition of f and g, so

$$h(x^1 \ldots x^{n'}) = f(g^1(x^1 \ldots x^{n'}), g^2(x^1 \ldots x^{n'}), \ldots g^n(z^1 \ldots x^{n'}))$$

Now assume that we would like to calculate the partial derivatives of h, namely

$$\frac{\partial h(x^1 \ldots x^{n'})}{\partial x^i} = \left.\frac{\partial y}{\partial x^i}\right|^{h}_{x^1 \ldots x^{n'}}$$

for $i = 1 \ldots n'$.

Then the chain rule is as follows:

$$\frac{\partial y}{\partial x^i} = \sum_{j=1}^{n} \frac{\partial y}{\partial z^j} \times \frac{\partial z^j}{\partial x^i} \tag{1.1}$$

or more precisely:

$$\left.\frac{\partial y}{\partial x^i}\right|^{h}_{x^1 \ldots x^{n'}} = \sum_{j=1}^{n} \left.\frac{\partial y}{\partial z^j}\right|^{f}_{z^1 \ldots z^n} \times \left.\frac{\partial z^j}{\partial x^i}\right|^{g^j}_{x^1 \ldots x^{n'}} \tag{1.2}$$

where $z^j = g^i(x^1 \ldots x^{n'})$ for $j = 1 \ldots n$.

Intuitively, the variable x^i affects each z^j through the function g^i; y is then effected by z^j through the function f. This is reflected through each term

$$\frac{\partial y}{\partial z^j} \times \frac{\partial z^j}{\partial x^i}$$

These contributions are summed to give the final value for $\frac{\partial y}{\partial x^i}$.

1.3.3 Jacobians

Now consider a function f that maps a sequence of *vectors* $z^1, z^2, \ldots z^n$ to a *vector* in $\mathbb{R}^m$:

$$y = f(z^1, z^2, \ldots z^n)$$

Assume in addition that each vector z^i is in $\mathbb{R}^{d^i}$: that is d^i specifies the dimensionality of z^i.

In this case for any $i \in \{1 \ldots n\}$ we use the notation

$$\frac{\partial f(z^1 \ldots z^n)}{\partial z^i}$$

or equivalently

$$\left.\frac{\partial y}{\partial z^i}\right|^{f}_{z^1\ldots z^n}$$

to refer to the *Jacobian matrix* of dimension $(m\times d^i)$, where for any $k\in\{1\ldots m\}$, $k'\in\{1\ldots d^i\}$,

$$\left[\frac{\partial f(z^1\ldots z^n)}{\partial z^i}\right]_{k,k'} = \left[\left.\frac{\partial y}{\partial z^i}\right|^{f}_{z^1\ldots z^n}\right]_{k,k'} = \left.\frac{\partial y_k}{\partial z^i_{k'}}\right|^{f_k}_{z^1\ldots z^n}$$

Thus the Jacobian

$$\left.\frac{\partial y}{\partial z^i}\right|^{f}_{z^1\ldots z^n}$$

contains a matrix of partial derivatives for each value y_k in the output paired with each value $z^i_{k'}$ in the input vector z^i. This Jacobian is a matrix of dimension $(m\times d^i)$, as there are m output values $y_1\ldots y_m$, and d^i input values in z^i, namely $z^i_1\ldots z^i_{d^i}$.

1.3.4 A Second Version of the Chain Rule

Again consider a function

$$y = f(z^1, z^2, \ldots z^n)$$

where y and $z^1\ldots z^n$ are vectors. Assume in addition that for $i=1\ldots n$

$$z^i = g^i(x^1\ldots x^{n'})$$

where $x^1\ldots x^{n'}$ are also vectors, and each g^i is a function.

This is very similar to the set-up in section 1.3.2, but where we have replaced scalars by vectors everywhere. We will see that this leads to a very similar form for the chain rule.

Define h to be the composition of f and g, so

$$h(x^1\ldots x^{n'}) = f(g^1(x^1\ldots x^{n'}), g^2(x^1\ldots x^{n'}), \ldots g^n(z^1\ldots x^{n'}))$$

Then the chain rule is as follows:

$$\underbrace{\frac{\partial y}{\partial x^j}}_{d(y)\times d(x^j)} = \sum_{i=1}^{n} \underbrace{\frac{\partial y}{\partial z^i}}_{d(y)\times d(z^i)} \times \underbrace{\frac{\partial z^i}{\partial x^j}}_{d(z^i)\times d(x^j)} \tag{1.3}$$

or more precisely:

$$\underbrace{\left.\frac{\partial y}{\partial x^j}\right|^{h}_{x_1 \ldots x_{n'}}}_{d(y)\times d(x^j)} = \sum_{i=1}^{n} \underbrace{\left.\frac{\partial y}{\partial z^i}\right|^{f}_{z_1 \ldots z_n}}_{d(y)\times d(z^i)} \times \underbrace{\left.\frac{\partial z^i}{\partial x^j}\right|^{g^i}_{x_1 \ldots x_{n'}}}_{d(z^i)\times d(x^j)} \tag{1.4}$$

Here we show matrix dimensions under each matrix; we use $d(v)$ for a vector v to denote the dimensionality of v.

This looks very similar to the chain rule in Eq. 1.2, where we have replaced scalars by vectors, and where the partial derivatives are now matrices corresponding to Jacobians.

Eq. 1.4 can in fact be derived using the rule in Eq. 1.2. In particular, by Eq. 1.2, for any output value y_k, and input variable $x^j_{k'}$, Eq. 1.2 implies that

$$\frac{\partial y_k}{\partial x^j_{k'}} = \sum_{i=1}^{n} \sum_{l=1}^{d(z^i)} \frac{\partial y_k}{\partial z^i_l} \times \frac{\partial z^i_l}{\partial x^j_{k'}}$$

It can be verified that Eq. 1.4 implements exactly this calculation for each k, k', through matrix multiplication of the Jacobians. It is an extremely convenient and compact form of the chain rule.

1.4 The Backpropagation Algorithm

Given a set of leaf values $u^1 \ldots u^l$, the forward algorithm calculates the output value u^n from the computational graph, implementing

$$u^n = h^n(u^1 \ldots u^l)$$

where h^n is the global function defined in section 1.2.4 of this note.

For any value $i \in \{1 \ldots l\}$, we would like to calculate the Jacobian

$$\underbrace{\frac{\partial u^n}{\partial u^i}}_{d(u^n)\times d(u^i)}$$

or more precisely

$$\underbrace{\left.\frac{\partial u^n}{\partial u^i}\right|^{h^n}_{u^1 \ldots u^l}}_{d(u^n)\times d(u^i)}$$

where the latter notation makes explicit that the partial derivative is being calculated with respect to the global function h^n. We will refer to this Jacobian as a *global Jacobian*, as it is based on the global function h^n.

This Jacobian summarizes the full set of partial derivatives

$$\frac{\partial u^n_k}{u^i_{k'}}$$

for $k \in \{1 \dots d(u^n)\}$ and $k' \in \{1 \dots d(u^i)\}$, where each u^n_k is a scalar corresponding to an output value, and each $u^i_{k'}$ is scalar corresponding to an input leaf value.

As a concrete example, in the computational graph in figure 1.4 corresponding to a feedforward neural network, the leaf variables are $x^i, y^i, W, b, V, \gamma$, and the output variable o is

$$o = -\log p(y^i|x^i; W, b, V, \gamma)$$

The global Jacobians

$$\underbrace{\frac{\partial o}{\partial W}}_{1\times(m\times d)}, \underbrace{\frac{\partial o}{\partial b}}_{1\times m}, \underbrace{\frac{\partial o}{\partial V}}_{1\times(K\times m)}, \underbrace{\frac{\partial o}{\partial \gamma}}_{1\times K}$$

contain the partial derivatives of o with respect to all parameters in the model.

Note that we have used the convention here that in terms such as

$$\frac{\partial o}{\partial W}$$

where one or both of the variables (in this case W) is a matrix, that matrices of dimension $(d \times d')$ are represented as vectors with $(d \times d')$ elements indexed by (i, j) where $1 \leq i \leq d$ and $1 \leq j \leq d'$. The Jacobian $\frac{\partial o}{\partial W}$ therefore has dimension $d(o) \times d(W)$, where W is represented as a vector with $m \times d$ elements.

1.4.1 The Local Jacobians

Recall that in the forward algorithm we compute values u^i for $i \in \{(l+1) \dots n\}$ using

$$u^i = f^i(\alpha^i)$$

where f^i is the local function associated with vertex i in the graph, and

$$\alpha^i = \langle u^j | (j, i) \in E \rangle$$

Given these definitions, for any edge (j, i) in the graph, we can calculate a *local Jacobian*. We will see that the local Jacobians play a central role in the backpropagation algorithm. They are defined as follows:

Definition 3 (Local Jacobian Functions) *For any edge $(j, i) \in E$, we define the* local Jacobian function $J^{j \to i}$ *as*

$$J^{j\to i}(\alpha^i) = \frac{\partial f^i(\alpha^i)}{\partial u^j} = \left.\frac{\partial u^i}{\partial u^j}\right|_{\alpha^i}^{f^i}$$

Note that $J^{j\to i}(\alpha^i)$ is a matrix of dimension $d^i \times d^j$. The input to the function α^i will generally be

$$\alpha^i = \langle u^j | (j, i) \in E\rangle$$

with $\alpha^i \in \mathbb{R}^{\bar{d}^i}$ where $\bar{d}^i = \sum_{j:(j,i)\in E} d^j$. It follows that $J^{j\to i}$ is a function in $\mathbb{R}^{\bar{d}^i} \to \mathbb{R}^{d^i \times d^j}$.

Next, assume that we have used the forward algorithm to calculate values $u^{l+1} \dots u^n$ given input leaf values $u^1 \dots u^l$. As before define

$$\alpha^i = \langle u^j | (j, i) \in E\rangle$$

Then for each edge $(j, i) \in E$, we will refer to the matrix

$$J^{j\to i}(\alpha^i) \in \mathbb{R}^{d^i \times d^j}$$

as the local Jacobian *for edge (j, i).* $\square$

Note that $J^{j\to i}(\alpha^i)$ is well defined if and only if the partial derivative $\frac{\partial f^i(\alpha^i)}{\partial u^j} = \left.\frac{\partial u^i}{\partial u^j}\right|_{\alpha^i}^{f^i}$ exists at point α^i. For now we will in general assume that all local Jacobians $J^{j\to i}(\alpha^i)$ are well defined; in section 1.4.4 we discuss the issue are Jacobians that are not defined due to non-differentiability of f^i with respect to some variable u^j.

Example 3 (An Example of Local Jacobians) *Consider again the example given earlier in the note, shown in figure 1.2. We have $n = 4$ and $l = 2$. We define the set of directed edges to be $E = \{(1, 3), (2, 3), (2, 4), (3, 4)\}$. We define $d^i = 1$ for $i = 1 \dots 4$: it follows that each variable u^i is a one-dimensional vector (i.e., a scalar). We define the local functions as follows:*

$$\begin{array}{rcl} f^3(u^1, u^2) & = & u^1 + u^2 \\ f^4(u^2, u^3) & = & u^2 \times u^3 \end{array}$$

The local Jacobian functions are then as follows:

$$J^{1\to 3}(u^1, u^2) = \left.\frac{\partial u^3}{\partial u^1}\right|_{u^1,u^2}^{f^3} = 1$$

$$J^{2\to 3}(u^1, u^2) = \left.\frac{\partial u^3}{\partial u^2}\right|_{u^1,u^2}^{f^3} = 1$$

$$J^{2\to 4}(u^2, u^3) = \left.\frac{\partial u^4}{\partial u^2}\right|_{u^2,u^3}^{f^4} = u^3$$

$$J^{3\to 4}(u^2, u^3) = \left.\frac{\partial u^4}{\partial u^3}\right|_{u^2,u^3}^{f^4} = u^2$$

Finally assume the leaf values are $u^1 = 2$ *and* $u^2 = 3$*. Then the forward algorithm proceeds with the following steps:*

$$u^3 = f^3(u^1, u^2) = u^1 + u^2 = 5$$

$$u^4 = f^4(u^2, u^3) = u^2 \times u^3 = 15$$

Now for each edge in the graph we can calculate the local Jacobian:

$$J^{1\to 3}(u^1, u^2) = 1$$

$$J^{2\to 3}(u^1, u^2) = 1$$

$$J^{2\to 4}(u^2, u^3) = u^3 = 5$$

$$J^{3\to 4}(u^2, u^3) = u^2 = 3$$

Note that in this example each variable u^i *in the graph has dimension* $d^i = 1$*; hence each Jacobian is a matrix of dimension* $(d^i \times d^j) = (1 \times 1)$*, i.e., each Jacobian is a scalar.*

1.4.2 The Backpropagation Algorithm

Figure 1.5 shows the backpropagation algorithm. The algorithm takes as input a computational graph, together with values for the leaf variables $u^1 \ldots u^l$. It returns as output the global Jacobians

$$\frac{\partial u^n}{\partial u^i}$$

or more precisely

$$\left.\frac{\partial u^n}{\partial u^i}\right|_{u^1\ldots u^l}^{h^n}$$

for $i \in \{1 \ldots l\}$.

The backward pass of the algorithm first initializes $P^n = I(d^n)$, where $I(d^n)$ is the identity matrix of dimension d^n. The algorithm then works backwards through the graph, for $j = (n-1)\dots 1$ calculating

$$P^j = \sum_{i:(j,i)\in E} P^i J^{j\to i}(\alpha^i)$$

The algorithm returns the values P^j for $j = 1\dots l$. We will prove in the next section that for $j = 1\dots l$,

$$P^j = \frac{\partial u^n}{\partial u^j}$$

or more precisely

$$P^j = \left.\frac{\partial u^n}{\partial u^j}\right|^{h^n}_{u^1\dots u^l}$$

Example 4 (An Example of the Backpropagation Algorithm) *Again computational graph in figure 1.2. We have $n = 4$ and $l = 2$. We define the set of directed edges to be $E = \{(1,3),(2,3),(2,4),(3,4)\}$. We define $d^i = 1$ for $i = 1\dots 4$: it follows that each variable u^i is a one-dimensional vector (i.e., a scalar). We define the local functions as follows:*

$$\begin{aligned} f^3(u^1,u^2) &= u^1 + u^2 \\ f^4(u^2,u^3) &= u^2 \times u^3 \end{aligned}$$

It follows that the local Jacobian functions are as follows:

$$\begin{aligned} J^{1\to 3}(u^1,u^2) &= 1 \\ J^{2\to 3}(u^1,u^2) &= 1 \\ J^{2\to 4}(u^2,u^3) &= u^3 \\ J^{3\to 4}(u^2,u^3) &= u^2 \end{aligned}$$

Now assume that we have leaf values $u^1 = 2$ and $u^2 = 3$. The forward algorithm calculates

$$u^3 = u^1 + u^2 = 5$$

$$u^4 = u^2 \times u^3 = 15$$

The backward pass then proceeds as follows:

$$\begin{aligned} P^4 &= 1 \quad \textit{(Initialization)} \\ P^3 &= P^4 \times J^{3\to 4}(u^2,u^3) = 1 \times u^2 = 3 \\ P^2 &= P^3 J^{2\to 3}(u^1,u^2) + P^4 J^{2\to 4}(u^2,u^3) = 3 \times 1 + 1 \times 5 = 8 \\ P^1 &= P^3 J^{1\to 3}(u^1,u^2) = 3 \times 1 = 3 \end{aligned}$$

Input: A computational graph $G = \langle n, l, E, d^1 \ldots d^n, u^1 \ldots u^n, f^1 \ldots f^n \rangle$. Values for the leaf variables $u^1 \ldots u^l$.

Algorithm (forward pass):

- For $i = (l+1) \ldots n$, compute

$$u^i = f^i(\alpha^i)$$

  where

$$\alpha^i = \langle u^j | (j,i) \in E \rangle$$

Algorithm (backward pass):

- Initialization: set $P^n = I(d^n)$ where $I(d^n)$ is the identity matrix of dimension $d^n \times d^n$.
- For $j = (n-1) \ldots 1$, compute

$$\underbrace{P^j}_{d^n \times d^j} = \sum_{i:(j,i) \in E} \underbrace{P^i}_{d^n \times d^i} \times \underbrace{J^{j \to i}(\alpha^i)}_{d^i \times d^j}$$

  where α^i is as calculated by the forward algorithm above.

Output: For $j = 1 \ldots l$, output the Jacobian

$$P^j = \frac{\partial u^n}{\partial u^j}\Bigg|_{u^1 \ldots u^l}^{h^n}$$

Figure 1.5: The backpropagation algorithm. The algorithm takes as inputs values for the leaf variables $u^1 \ldots u^l$, and returns the partial derivatives $\frac{\partial u^n}{\partial u^j}$ for $j \in \{1 \ldots l\}$. The algorithm makes use of a forward pass followed by a backward pass.

The outputs from the algorithm are

$$\frac{\partial u^4}{\partial u^1}\Big|_{u^1 \ldots u^2}^{h^n} = P^1 = 3$$

$$\frac{\partial u^4}{\partial u^2}\Big|_{u^1 \ldots u^2}^{h^n} = P^2 = 8$$

We can verify that these values are correct as follows. The global functions h^3 and h^4 can be calculated as

$$h^3(u^1, u^2) = u^1 + u^2$$

$$h^4(u^1, u^2) = u^2 \times h^3(u^1, u^2) = u^1 \times u^2 + (u^2)^2$$

It follows that

$$\frac{\partial h^4(u^1, u^2)}{\partial u^1} = u^2 = 3$$

$$\frac{\partial h^4(u^1, u^2)}{\partial u^2} = u^1 + 2 \times u^2 = 8$$

which matches the values calculated by the algorithm.

1.4.3 Justification for the Backpropagation Algorithm

We now give justification for the backpropagation algorithm. We first introduce the following definitions:

Definition 4 *Directed Paths. Given a computational graph, we define $\mathcal{P}(j, i)$ to be the set of* directed paths *between j and i in the graph. each directed path is a sequence of edges $(v_1, v_2), \ldots, (v_{n-1}, v_n)$ where $v_1 = j$, $v_n = i$, $n \geq 2$, and $(v_i, v_{i+1}) \in E$ for all i.*

Definition 5 *Products of Jacobians over Directed Paths. For a path $p = (v_1, v_2), \ldots, (v_{n-1}, v_n)$ we define*

$$\prod_{(a,b) \in p} J^{a \to b}(\alpha^b) = J^{v_{n-1} \to v_n}(\alpha^{v_n}) \times J^{v_{n-2} \to v_{n-1}}(\alpha^{v_{n-1}}) \ldots \times J^{v_1 \to v_2}(\alpha^{v_2})$$

Hence this corresponds to the product of Jacobians over the path, where the first term in the product is $J^{v_{n-1} \to v_n}(\alpha^{v_n})$, and the last term in the product is $J^{v_1 \to v_2}(\alpha^{v_2})$.

Note that for matrix multiplications the order of products is important—in general for two matrices X and Y, $X \times Y \neq Y \times X$ because matrix multiplication is not commutative—hence it is important to specify the order in this definition.

We can now state the following theorem:

Theorem 1 *Consider a computational graph $\langle n, l, E, u^1 \dots u^n, d^1 \dots d^n, f^1 \dots f^n \rangle$. Assume we have values for the leaf variables $u^1 \dots u^l$ and we use the forward algorithm to calculate values for $u^{l+1} \dots u^n$. As usual define*

$$\alpha^i = \langle u^j | (j, i) \in E \rangle$$

Assume that for every edge $(j, i) \in E$, the local Jacobian

$$J^{j \to i}(\alpha^i) = \frac{\partial f^i(\alpha^i)}{\partial u^j} = \left. \frac{\partial u^i}{\partial u^j} \right|_{\alpha^i}^{f^i}$$

is well defined.

Then for any $j \in \{1 \dots l\}$, $i \in \{(l+1) \dots n\}$, we have

$$\left. \frac{\partial u^i}{\partial u^j} \right|_{u^1 \dots u^l}^{h^i} = \sum_{p \in \mathcal{P}(j,i)} \prod_{(a,b) \in p} J^{a \to b}(\alpha^b)$$

where $\mathcal{P}(j, i)$ is the set of directed paths between vertex j and vertex i in the graph; each directed path is a sequence of edges $(v_1, v_2), \dots, (v_{n-1}, v_n)$ where $v_1 = j$, $v_n = i$, $n \geq 2$, and $(v_i, v_{i+1}) \in E$ for all i.

Thus to calculate the partial derivative

$$\left. \frac{\partial u^i}{\partial u^j} \right|_{u^1 \dots u^l}^{h^i}$$

for $i \in \{(l+1) \dots n\}$ and $j \in \{1 \dots l\}$, we sum over all directed paths between j and i in the graph, and take the product of Jacobians along each path.

The proof is given in section 1.4.5: it is a direct application of the chain rule, together with proof by induction, to the computational graph.

Example 5 *As an example, consider again a computational graph with leaf variables u^1, u^2, and edges $E = \{(1, 3), (2, 3), (2, 4), (3, 4)\}$. Consider the calculation of*

$$\left. \frac{\partial u^4}{\partial u^2} \right|_{u^1 \dots u^2}^{h^4}$$

In this case there are two directed paths between vertex 2 and 4 in the graph:

$$p_1 = (2, 3)(3, 4)$$

$$p_2 = (2, 4)$$

It follows that

$$\left.\frac{\partial u^4}{\partial u^2}\right|_{u^1 \ldots u^2}^{h^4} = J^{3\to 4}(u^2, u^3) \times J^{2\to 3}(u^1, u^2) + J^{2\to 4}(u^1, u^2)$$

For example with $u^1 = 2$ *and* $u^2 = 3$ *we have* $u^3 = 5$ *and* $u^4 = 15$*, and*

$$\left.\frac{\partial u^4}{\partial u^2}\right|_{u^1 \ldots u^2}^{h^4} = 3 \times 1 + 5 = 8$$

The second theorem completes the justification for the backpropagation algorithm:

Theorem 2 *Consider the backpropagation algorithm in figure 1.5. For* $j = (n - 1) \ldots 1$ *the following holds:*

$$P^j = \sum_{p \in \mathcal{P}(j,n)} \prod_{(a,b) \in p} J^{a\to b}(\alpha^b)$$

It follows from theorem 1 that for $j = 1 \ldots l$*,*

$$P^j = \left.\frac{\partial u^n}{\partial u^j}\right|_{u^1 \ldots u^l}^{h^n}$$

The proof is by induction, and is given in section 1.4.6. The algorithm in figure 1.5 is a dynamic programming algorithm, which calculates

$$P^j = \sum_{p \in \mathcal{P}(j,n)} \prod_{(a,b) \in p} J^{a\to b}(\alpha^b)$$

for every vertex j in the graph.

1.4.4 An Extension to the Case where Some Local Jacobians are not Defined

Theorem 1 assumes that each local Jacobian

$$J^{j\to i}(\alpha^i) = \frac{\partial f^i(\alpha^i)}{\partial u^j} = \left.\frac{\partial u^i}{\partial u^j}\right|_{\alpha^i}^{f^i}$$

in the graph is well defined. We now consider cases where some local Jacobians are not well defined, due to the partial derivative

$$\frac{\partial f^i(\alpha^i)}{\partial u^j}$$

not being well defined at the point α^i.

As one example, consider the computational graph for a feedfoward neural network in figure 1.4, where we have

$$o = -q_{y^i}$$

or equivalently

$$u^{11} = -u^{10}_{u^4}$$

where $u^4 = y^i$ is an integer in $\{1, 2, \ldots K\}$ specifying the output label of the model.

Because u^4 is a discrete (as opposed to continous) variable, the partial derivative

$$\left.\frac{\partial u^{11}}{\partial u^4}\right|^{f^{11}}_{\alpha^{11}}$$

is not well defined.

Intuitively, if we are trying to calculate the partial derivative

$$\left.\frac{\partial u^n}{\partial u^j}\right|^{f^n}_{u^1 \ldots u^l}$$

for some leaf variable $j \in \{1 \ldots l\}$, and there is some directed path from j to n such that a local Jacobian on that path is not well defined, then the partial derivative $\left.\frac{\partial u^n}{\partial u^j}\right|^{f^n}_{u^1 \ldots u^l}$ will not be well defined.

This leads us to the following definition:

Definition 6 *For any computational graph $\langle n, l, E, u^1 \ldots u^n, d^1 \ldots d^n, f^1 \ldots f^n \rangle$, for any leaf values $u^1 \ldots u^l$, we define the set*

$$\mathcal{D}^i(u^1 \ldots u^l) \subseteq \{1 \ldots l\}$$

to be the set of values $j \in \{1 \ldots l\}$, such that for every path $p \in \mathcal{P}(j, i)$, for every edge $(a, b) \in p$, the Jacobian

$$J^{a \rightarrow b}(\alpha^b)$$

is well defined, where α^b is calculated using the forward algorithm with input values $u^1 \ldots u^l$.

We can then state a modified version of theorem 1:

Theorem 3 *Consider a computational graph $\langle n, l, E, u^1 \ldots u^n, d^1 \ldots d^n, f^1 \ldots f^n \rangle$. Assume we have values for the leaf variables $u^1 \ldots u^l$ and we use the forward algorithm to calculate values for $u^{l+1} \ldots u^n$. As usual define*

$$\alpha^i = \langle u^j | (j,i) \in E \rangle$$

Then for any $j \in \{1 \ldots l\}$, $i \in \{(l+1) \ldots n\}$ such that $j \in \mathcal{D}^i(u^1 \ldots u^l)$ we have

$$\left. \frac{\partial u^i}{\partial u^j} \right|_{u^1 \ldots u^l}^{h^i} = \sum_{p \in \mathcal{P}(j,i)} \prod_{(a,b) \in p} J^{a \to b}(\alpha^b)$$

where $\mathcal{P}(j,i)$ is the set of directed paths between vertex j and vertex i in the graph; each directed path is a sequence of edges $(v_1, v_2), \ldots, (v_{n-1}, v_n)$ where $v_1 = j$, $v_n = i$, $n \geq 2$, and $(v_i, v_{i+1}) \in E$ for all i.

Thus we have simply modified the theorem to apply only to pairs of vertices (j,i) where all directed paths from j to i have well-defined local Jacobians. The proof of the theorem is very similar to the original proof.

1.4.5 Proof of Theorem 1

The proof is by induction. Assume we have leaf values $u^1 \ldots u^l$ and we have used the forward algorithm to calculate values for $u^{l+1} \ldots u^n$. As usual define

$$\alpha^i = \langle u^j | (j,i) \in E \rangle$$

For convenience, for any edge $(j,i) \in E$, define

$$D^{j \to i} = J^{j \to i}(\alpha^i)$$

As before, define $\mathcal{P}(j,i)$ to be the set of directed paths from j to i in the graph. Each directed path is a sequence of edges $(v_1, v_2) \ldots (v_{n-1}, v_n)$ where $n \geq 2$, each $(v_i, v_{i+1}) \in E$, and $v_1 = j$, $v_n = i$.

In addition, define $\mathcal{P}(j,k,i)$ to be the set of directed paths from j to i in the graph, with the final edge equal to (k,i): i.e., each path in $\mathcal{P}(j,k,i)$ is a sequence of edges $(v_1, v_2) \ldots (v_{n-1}, v_n)$ where $n \geq 2$, each $(v_i, v_{i+1}) \in E$, $v_1 = j$, $v_n = i$, and $v_{n-1} = k$. Note that $\mathcal{P}(j,j,i)$ is equal to (j,i) if $(j,i) \in E$, and is empty otherwise.

It follows that

$$\mathcal{P}(j,i) = \cup_{k:(k,i) \in E} \mathcal{P}(j,k,i)$$

and for k, k' such that $k \neq k'$,

$$\mathcal{P}(j,k,i) \cap \mathcal{P}(j,k',i) = \emptyset$$

where $\emptyset$ is the empty set.

Recall the definition of the global functions h^i for $i = 1 \ldots n$:

- If $i \in \{1 \ldots l\}$
$$h^i(u^1 \ldots u^l) = u^i$$

- else if $i \in (l+1) \ldots n$,
$$h^i(u^1 \ldots u^l) = f^i(\alpha^i)$$
where
$$\alpha^i = \langle h^j(u^1 \ldots u^l) | (j,i) \in E \rangle$$

It follows by the chain rule, for any $i \in \{(l+1) \ldots n\}$, $j \in \{1 \ldots l\}$,

$$\frac{\partial u^i}{\partial u^j}\bigg|_{u^1 \ldots u^l}^{h^i} = \sum_{k:(k,i)\in E} \frac{\partial u^i}{\partial u^k}\bigg|_{\alpha^i}^{f^i} \times \frac{\partial u^k}{\partial u^j}\bigg|_{u^1 \ldots u^l}^{h^k} \quad (1.5)$$

$$= \sum_{k:(k,i)\in E} D^{k\to i} \times \frac{\partial u^k}{\partial u^j}\bigg|_{u^1 \ldots u^l}^{h^k} \quad (1.6)$$

where in the last equality we have used

$$D^{k\to i} = J^{k\to i}(\alpha^i) = \frac{\partial u^i}{\partial u^k}\bigg|_{\alpha^i}^{f^i}$$

Now consider the expression

$$D^{k\to i} \times \frac{\partial u^k}{\partial u^j}\bigg|_{u^1 \ldots u^l}^{h^k}$$

There are three cases:

- $k \in \{1 \ldots l\}$, $k \neq j$: in this case
$$\frac{\partial u^k}{\partial u^j}\bigg|_{u^1 \ldots u^l}^{h^k} = 0$$

hence

$$\begin{aligned} D^{k\rightarrow i} \times \left.\frac{\partial u^k}{\partial u^j}\right|^{h^k}_{u^1\ldots u^l} &= 0 \\ &= \sum_{p\in\mathcal{P}(j,k,i)} \prod_{(a,b)\in p} D^{a\rightarrow b} \end{aligned}$$

where the last equality follows because $\mathcal{P}(j,k,i)$ is the empty set if $k \in \{1\ldots l\}$, $k \neq j$.

- $k \in \{1\ldots l\}$, $k = j$: in this case

$$\left.\frac{\partial u^k}{\partial u^j}\right|^{h^k}_{u^1\ldots u^l} = I(d^j)$$

where $I(d^j)$ is the identity matrix of dimension $d^j \times d^j$, hence

$$\begin{aligned} D^{k\rightarrow i} \times \left.\frac{\partial u^k}{\partial u^j}\right|^{h^k}_{u^1\ldots u^l} &= D^{k\rightarrow i} \\ &= \sum_{p\in\mathcal{P}(j,k,i)} \prod_{(a,b)\in p} D^{a\rightarrow b} \end{aligned}$$

because with $k \in \{1\ldots l\}$, $k = j$, the set $\mathcal{P}(j,k,i)$ contains a single path, which contains the single edge $(k,i) = (j,i)$.

- $k \in \{(l+1)\ldots n\}$: in this case by the inductive hypothesis,

$$\left.\frac{\partial u^k}{\partial u^j}\right|^{h^k}_{u^1\ldots u^l} = \sum_{p\in\mathcal{P}(j,k)} \prod_{(a,b)\in p} D^{a\rightarrow b}$$

It follows that

$$\begin{aligned} D^{k\rightarrow i} \times \left.\frac{\partial u^k}{\partial u^j}\right|^{h^k}_{u^1\ldots u^l} &= D^{k\rightarrow i} \times \sum_{p\in\mathcal{P}(j,k)} \prod_{(a,b)\in p} D^{a\rightarrow b} \\ &= \sum_{p\in\mathcal{P}(j,k)} D^{k\rightarrow i} \times \prod_{(a,b)\in p} D^{a\rightarrow b} \\ &= \sum_{p\in\mathcal{P}(j,k,i)} \prod_{(a,b)\in p} D^{a\rightarrow b} \end{aligned}$$

We have shown that for all k such that $(k,i) \in E$,

$$D^{k\to i} \times \frac{\partial u^k}{\partial u^j}\bigg|^{h^k}_{u^1\dots u^l} = \sum_{p\in\mathcal{P}(j,k,i)} \prod_{(a,b)\in p} D^{a\to b}$$

It follows from Eq. 1.6 that

$$\begin{aligned}\frac{\partial u^i}{\partial u^j}\bigg|^{h^i}_{u^1\dots u^l} &= \sum_{k:(k,i)\in E} D^{k\to i} \times \frac{\partial u^k}{\partial u^j}\bigg|^{h^k}_{u^1\dots u^l} \\ &= \sum_{k:(k,i)\in E} \sum_{p\in\mathcal{P}(j,k,i)} \prod_{(a,b)\in p} D^{a\to b} \\ &= \sum_{p\in\mathcal{P}(j,i)} \prod_{(a,b)\in p} D^{a\to b} \qquad (1.7)\end{aligned}$$

where the final equality follows from

$$\mathcal{P}(j,i) = \cup_{k:(k,i)\in E}\mathcal{P}(j,k,i)$$

and for k, k' such that $k \neq k'$,

$$\mathcal{P}(j,k,i) \cap \mathcal{P}(j,k',i) = \emptyset$$

□

1.4.6 Proof of Theorem 2

The proof is by induction. Assume we have leaf values $u^1 \dots u^l$ and we have used the forward algorithm to calculate values for $u^{l+1} \dots u^n$. We make use of the following definitions:

- As usual define
$$\alpha^i = \langle u^j | (j,i) \in E\rangle$$
For convenience, for any edge $(j,i) \in E$, define
$$D^{j\to i} = J^{j\to i}(\alpha^i)$$

- As before, define $\mathcal{P}(j,i)$ to be the set of directed paths from j to i in the graph. Each directed path is a sequence of edges $(v_1, v_2)\dots(v_{n-1}, v_n)$ where $n \geq 2$, each $(v_i, v_{i+1)} \in E$, and $v_1 = j$, $v_n = i$.

- In addition, define $\mathcal{P}'(j,k,i)$ to be the set of directed paths from j to i in the graph, with the *first* edge equal to (j,k): i.e., each path in $\mathcal{P}'(j,k,i)$ is a sequence of edges $(v_1,v_2)\ldots(v_{n-1},v_n)$ where $n \geq 2$, each $(v_i, v_{i+1)} \in E$, $v_1 = j, v_2 = k$, and $v_n = i$. Note that $\mathcal{P}(j,i,i)$ is equal to (j,i) if $(j,i) \in E$, and is empty otherwise.

  It follows that
  $$\mathcal{P}(j,n) = \cup_{i:(j,i)\in E}\mathcal{P}(j,i,n)$$
  and for i, i' such that $i \neq i'$,
  $$\mathcal{P}(j,i,n) \cap \mathcal{P}(j,i',n) = \emptyset$$
  where $\emptyset$ is the empty set.

The backpropagation algorithm calculates values for P^j for $j = (n-1)\ldots 1$ as follows:

$$P^j \quad = \sum_{i:(j,i)\in E} P^i \times D^{j\to i} \tag{1.8}$$

where we have used $D^{j\to i} = J^{j\to i}(\alpha^i)$.

Now consider the term
$$P^i \times D^{j\to i}$$

There are two cases:

- $i = n$. In this case $P^i = I(d^n)$ where $I(d^n)$ is the identity matrix of dimension $d^n \times d^n$. It follows that
  $$\begin{aligned} P^i \times D^{j\to i} &= D^{j\to i} \\ &= \sum_{p\in\mathcal{P}'(j,i,n)} \prod_{(a,b)\in p} D^{a\to b} \end{aligned}$$
  where the last equality follows because the set $\mathcal{P}'(j,i,n)$ for $i = n$ contains a single path, which consists of a single edge $(j,n) = (j,i)$.

- $i < n$. In this case by the inductive hypothesis,
  $$P^i = \sum_{p\in\mathcal{P}(i,n)} \prod_{(a,b)\in p} D^{a\to b}$$
  It follows that
  $$\begin{aligned} P^i \times D^{j\to i} &= \left(\sum_{p\in\mathcal{P}(i,n)} \prod_{(a,b)\in p} D^{a\to b}\right) \times D^{j\to i} \\ &= \sum_{p\in\mathcal{P}'(j,i,n)} \prod_{(a,b)\in p} D^{a\to b} \end{aligned}$$

We have shown above that for all i such that $(j, i) \in E$,

$$P^i \times D^{j \to i} = \sum_{p \in \mathcal{P}'(j,i,n)} \prod_{(a,b) \in p} D^{a \to b}$$

Combining this with Eq.1.8 gives

$$\begin{aligned} P^j &= \sum_{i:(j,i) \in E} P^i \times D^{j \to i} \\ &= \sum_{i:(j,i) \in E} \sum_{p \in \mathcal{P}'(j,i,n)} \prod_{(a,b) \in p} D^{a \to b} && (1.9) \\ &= \sum_{p \in \mathcal{P}(j,n)} \prod_{(a,b) \in p} D^{a \to b} && (1.10) \end{aligned}$$

which completes the proof. $\square$