Automatic Differentiation: An Overview of Forward and Reverse Mode in Applications to Optimization Problems

Who? Bent Müller

From? University of Hamburg

When? 17.05.2022

What is automatic differentiation?

Given a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ that is *twice differentiable*, we want to *efficiently* and with good *numerical stability* compute

- $f(x)$, the value of our function at a point x in $\mathbb{R}^n$,
- $\nabla f(x)$, the gradient of our function at x, and
- $\nabla^2 f(x)$, the Hessian of our function at x.

Where ∇ is the *Differential Operator*, also called the *nabla* Operator or sometimes just *Del*.

Quick reminder of the symbols

Since $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, we know that $f(x) \in \mathbb{R}$, now we define the Gradient and Hessian for $x \in O \subset \mathbb{R}^n$ where O is an open subset in $\mathbb{R}^n$ as follows:

Quick reminder of the symbols

Since $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, we know that $f(x) \in \mathbb{R}$, now we define the Gradient and Hessian for $x \in O \subset \mathbb{R}^n$ where O is an open subset in $\mathbb{R}^n$ as follows:

$$\nabla f(x) := \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{pmatrix}$$

The Gradient is the n-dimensional *vector* of partial derivatives of f at x. It describes how f changes with respect to each of the n variables $x_1, \ldots, x_n$.

Quick reminder of the symbols

Since $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, we know that $f(x) \in \mathbb{R}$, now we define the Gradient and Hessian for $x \in O \subset \mathbb{R}^n$ where O is an open subset in $\mathbb{R}^n$ as follows:

$$\nabla\left(\nabla f(x)\right) = \nabla^2 f(x) := \begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{pmatrix}$$

The Hessian is the real matrix of all *second order* partial derivatives of f at x. It describes how the gradient changes with respect to each of the n variables $x_1, \ldots, x_n$.

Quick reminder of the symbols

Since $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, we know that $f(x) \in \mathbb{R}$, now we define the Gradient and Hessian for $x \in O \subset \mathbb{R}^n$ where O is an open subset in $\mathbb{R}^n$ as follows:

$$\nabla\left(\nabla f(x)\right) = \nabla^2 f(x) := \begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{pmatrix}$$

Each row of the Hessian can be regarded as the gradient w.r.t. each component of the gradient vector of f. Specifically, it is the *Jacobian* matrix of the gradient.

Characterizing Sequence

We will only discuss functions that can be written as compositions of:

- constant functions C,
- unary functions U and
- binary functions B.

Where C, U and B are the corresponding function spaces.

The characterizing sequence computes $f(x)$ in m-steps, with each step depending *only* on previously computed steps.

Characterizing Sequence

We decompose the computation of $f(x)$ as follows:

- $f_i = x_i$ for $i \in \{1, \ldots, n\}$
- $f_{i+n} = \begin{cases} \omega_i & \text{if } \omega_i \in \mathcal{C} \\ \omega_i(f_{k_i}) & \text{if } \omega_i \in \mathcal{U} \\ \omega_i(f_{k_i}, f_{l_i}) & \text{if } \omega_i \in \mathcal{B} \end{cases} \qquad \text{for } i \in \{1, \ldots, m\}$
- $f_{m+n} = f(x)$
- $k_i, l_i < i + n$ and
- $\{n+1, \ldots, n+m-1\} \subset \bigcup_{i=1}^{m} \{k_i, l_i\}$

The last condition ensures that each of the m-steps in the computation is actually used to compute $f(x)$.

$$\Rightarrow S := ((\omega_i, k_i, l_i))_{i \in \{1, \ldots, m\}} \text{ is char. seq. for f}$$

that is defined in the following.

Characterizing Sequence

More specifically, we set $I := \{1, \ldots, m\}$ and $J := \{1, \ldots, n+m-1\}$ as two sets of indices.

$$\begin{aligned}\Rightarrow S &:= ((\omega_i, k_i, l_i))_{i\in\{1,\ldots,m\}} \\ &\in \left(\left(\mathcal{C} \times \{0\}^2\right) \cup \left(\mathcal{U} \times J \times \{0\}\right) \cup \left(\mathcal{B} \times J^2\right)\right)^m\end{aligned}$$

The above is equivalent to the following three cases:

Characterizing Sequence

More specifically, we set $I := \{1, \dots, m\}$ and $J := \{1, \dots, n+m-1\}$ as two sets of indices.

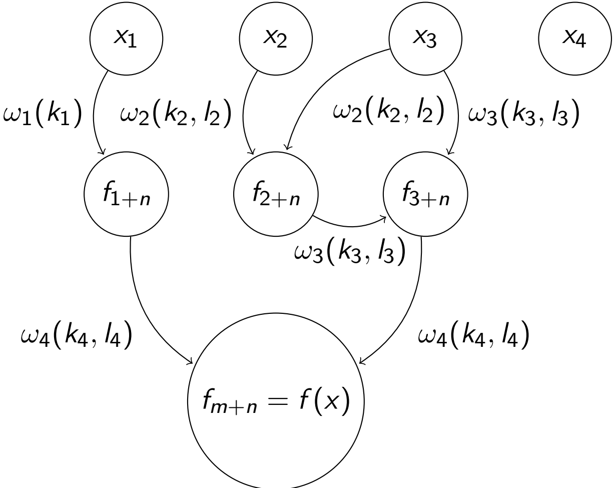
$$\begin{aligned}\Rightarrow S &:= ((\omega_i, k_i, l_i))_{i \in \{1,\dots,m\}} \\ &\in \left(\left(\mathcal{C} \times \{0\}^2\right) \cup \left(\mathcal{U} \times J \times \{0\}\right) \cup \left(\mathcal{B} \times J^2\right)\right)^m\end{aligned}$$

The above is equivalent to the following three cases:

- if $\omega_i \in \mathcal{C} \Rightarrow k_i = l_i = 0$
- if $\omega_i \in \mathcal{U} \Rightarrow k_i \in J, l_i = 0$
- if $\omega_i \in \mathcal{B} \Rightarrow k_i, l_i \in J$

The Computational Graph

An example computational graph constructed by a characterizing sequence for $n = 4$ and $m = 4$.



Computing the Gradient $\nabla f(x)$

The idea is to compute the gradient and hessian of f stepwise using the characterizing sequence S.

Computing the Gradient $\nabla f(x)$

- $g_j = e_j$ with $(e_j)_k = 1_{k=j}$ and $e_j \in \mathbb{R}^n$ for $j \in \{1, \ldots, n\}$
- $$g_{i+n} = \begin{cases} 0 & \text{if } \omega_i \in \mathcal{C} \\ \omega_i'(f_{k_i}) g_{k_i} & \text{if } \omega_i \in \mathcal{U} \\ \partial_a \omega_i(f_{k_i}, f_{l_i}) g_{k_i} + \partial_b \omega_i(f_{k_i}, f_{l_i}) g_{l_i} & \text{if } \omega_i \in \mathcal{B} \end{cases},$$
  for $i \in \{1, \ldots, m\}$

  Note the following:

Computing the Gradient $\nabla f(x)$

- $g_j = e_j$ with $(e_j)_k = 1_{k=j}$ and $e_j \in \mathbb{R}^n$ for $j \in \{1, \dots, n\}$
- $$g_{i+n} = \begin{cases} 0 & \text{if } \omega_i \in \mathcal{C} \\ \omega_i'(f_{k_i}) g_{k_i} & \text{if } \omega_i \in \mathcal{U} \\ \partial_a \omega_i(f_{k_i}, f_{l_i}) g_{k_i} + \partial_b \omega_i(f_{k_i}, f_{l_i}) g_{l_i} & \text{if } \omega_i \in \mathcal{B} \end{cases},$$
  for $i \in \{1, \dots, m\}$

Note the following:

The g_i are n-dimensional vectors $\iff g_i \in \mathbb{R}^n$

Computing the Gradient $\nabla f(x)$

- $g_j = e_j$ with $(e_j)_k = 1_{k=j}$ and $e_j \in \mathbb{R}^n$ for $j \in \{1, \ldots, n\}$
- $$g_{i+n} = \begin{cases} 0 & \text{if } \omega_i \in \mathcal{C} \\ \omega_i'(f_{k_i}) g_{k_i} & \text{if } \omega_i \in \mathcal{U} \\ \partial_a \omega_i(f_{k_i}, f_{l_i}) g_{k_i} + \partial_b \omega_i(f_{k_i}, f_{l_i}) g_{l_i} & \text{if } \omega_i \in \mathcal{B} \end{cases},$$
for $i \in \{1, \ldots, m\}$

Note the following:

For $\omega_i \in \mathcal{U}$ we have:

$$\omega_i'(f_{k_i}) \in \mathbb{R} \text{ since } \omega_i : \mathbb{R} \to \mathbb{R}$$
$$\Rightarrow \omega_i'(f_{k_i}) g_{k_i} \in \mathbb{R}^n$$

Computing the Gradient $\nabla f(x)$

- $g_j = e_j$ with $(e_j)_k = 1_{k=j}$ and $e_j \in \mathbb{R}^n$ for $j \in \{1, \dots, n\}$
- $$g_{i+n} = \begin{cases} 0 & \text{if } \omega_i \in \mathcal{C} \\ \omega_i'(f_{k_i}) g_{k_i} & \text{if } \omega_i \in \mathcal{U} \\ \partial_a \omega_i(f_{k_i}, f_{l_i}) g_{k_i} + \partial_b \omega_i(f_{k_i}, f_{l_i}) g_{l_i} & \text{if } \omega_i \in \mathcal{B} \end{cases},$$

for $i \in \{1, \dots, m\}$

Note the following:

For $\omega_i \in \mathcal{B}$ we have:

Applying the chain rule, we get:

$$\begin{aligned} \nabla \omega_i \left(f_{k_i}, f_{l_i}\right) &:= \left(\frac{\partial \left(\omega_i \left(f_{k_i}, f_{l_i}\right)\right)}{\partial x_1}, \dots, \frac{\partial \left(\omega_i \left(f_{k_i}, f_{l_i}\right)\right)}{\partial x_n} \right)^T \\ &= \frac{\partial \left(\omega_i \left(f_{k_i}, f_{l_i}\right)\right)}{\partial f_{k_i}} \nabla f_{k_i} + \frac{\partial \left(\omega_i \left(f_{k_i}, f_{l_i}\right)\right)}{\partial f_{l_i}} \nabla f_{l_i} \end{aligned}$$

Computing the Gradient $\nabla f(x)$

- $g_j = e_j$ with $(e_j)_k = 1_{k=j}$ and $e_j \in \mathbb{R}^n$ for $j \in \{1, \dots, n\}$
- $$g_{i+n} = \begin{cases} 0 & \text{if } \omega_i \in \mathcal{C} \\ \omega_i'(f_{k_i}) g_{k_i} & \text{if } \omega_i \in \mathcal{U} \\ \partial_a \omega_i(f_{k_i}, f_{l_i}) g_{k_i} + \partial_b \omega_i(f_{k_i}, f_{l_i}) g_{l_i} & \text{if } \omega_i \in \mathcal{B} \end{cases},$$
  for $i \in \{1, \dots, m\}$

Note the following:

For $\omega_i \in \mathcal{B}$ we have:

$$\begin{aligned} \Rightarrow \nabla \omega_i(f_{k_i}, f_{l_i}) &= \partial_a \omega_i(f_{k_i}, f_{l_i}) \nabla f_{k_i} + \partial_b \omega_i(f_{k_i}, f_{l_i}) \nabla f_{l_i} \\ &= \partial_a \omega_i(f_{k_i}, f_{l_i}) g_{k_i} + \partial_b \omega_i(f_{k_i}, f_{l_i}) g_{l_i} \end{aligned}$$

With $\partial_a \omega_i(f_{k_i}, f_{l_i})$ being the partial derivative in the *first* argument, that is w.r.t. f_{k_i}. Note that ∇ denotes the *total derivative* in this context.

Computing the Gradient $\nabla f(x)$

- $g_j = e_j$ with $(e_j)_k = 1_{k=j}$ and $e_j \in \mathbb{R}^n$ for $j \in \{1, \dots, n\}$
- $$g_{i+n} = \begin{cases} 0 & \text{if } \omega_i \in \mathcal{C} \\ \omega_i'(f_{k_i}) g_{k_i} & \text{if } \omega_i \in \mathcal{U} \\ \partial_a \omega_i(f_{k_i}, f_{l_i}) g_{k_i} + \partial_b \omega_i(f_{k_i}, f_{l_i}) g_{l_i} & \text{if } \omega_i \in \mathcal{B} \end{cases},$$
  for $i \in \{1, \dots, m\}$

Note the following:

Thus, we have for $i \in \{1, \dots, n+m\}$:

$$g_i = \nabla f_i = \begin{pmatrix} \frac{\partial f_i}{\partial x_1} \\ \frac{\partial f_i}{\partial x_2} \\ \vdots \\ \frac{\partial f_i}{\partial x_n} \end{pmatrix}$$

Computing the Gradient $\nabla f(x)$

- $g_j = e_j$ with $(e_j)_k = 1_{k=j}$ and $e_j \in \mathbb{R}^n$ for $j \in \{1, \dots, n\}$
- $$g_{i+n} = \begin{cases} 0 & \text{if } \omega_i \in \mathcal{C} \\ \omega_i'(f_{k_i}) g_{k_i} & \text{if } \omega_i \in \mathcal{U} \\ \partial_a \omega_i(f_{k_i}, f_{l_i}) g_{k_i} + \partial_b \omega_i(f_{k_i}, f_{l_i}) g_{l_i} & \text{if } \omega_i \in \mathcal{B} \end{cases},$$
  for $i \in \{1, \dots, m\}$

Note the following:

So for all $i \in \{1, \dots, n+m\}$, g_i is really the gradient vector of f_i w.r.t. all inputs $(x_1, \dots, x_n)$.

$\Rightarrow$ Auto-Diff applies the chain rule iteratively.

Computing the Hessian $\nabla^2 f(x)$

- $H_j = 0 \in \mathbb{R}^{n\times n}$ for $j \in \{1, \dots, n\}$
- $$H_{i+n} = \begin{cases} 0 & \text{if } \omega_i \in \mathcal{C} \\ \omega_i'(f_{k_i}) H_{k_i} + \omega_i''(f_{k_i}) g_{k_i} g_{k_i}^T & \text{if } \omega_i \in \mathcal{U} \\ \partial_a \omega_i(f_{k_i}, f_{l_i}) H_{k_i} + \partial_b \omega_i(f_{k_i}, f_{l_i}) H_{l_i} & \\ +(g_{k_i}, g_{l_i}) \nabla^2 \omega_i(f_{k_i}, f_{l_i}) (g_{k_i}, g_{l_i})^T & \text{if } \omega_i \in \mathcal{B} \end{cases}$$

Note that $g_{k_i} g_{k_i}^T$ is a $n \times n$ matrix (outer product) , while $g_{k_i}^T g_{k_i}$ is only a single number (dot product).

Computing the Hessian $\nabla^2 f(x)$

- $H_j = 0 \in \mathbb{R}^{n\times n}$ for $j \in \{1, \dots, n\}$
- $$H_{i+n} = \begin{cases} 0 & \text{if } \omega_i \in \mathcal{C} \\ \omega_i'(f_{k_i}) H_{k_i} + \omega_i''(f_{k_i}) g_{k_i} g_{k_i}^T & \text{if } \omega_i \in \mathcal{U} \\ \partial_a \omega_i(f_{k_i}, f_{l_i}) H_{k_i} + \partial_b \omega_i(f_{k_i}, f_{l_i}) H_{l_i} & \\ +(g_{k_i}, g_{l_i}) \nabla^2 \omega_i(f_{k_i}, f_{l_i}) (g_{k_i}, g_{l_i})^T & \text{if } \omega_i \in \mathcal{B} \end{cases}$$

Now assuming $a : \mathbb{R}^n \to \mathbb{R}$ and $b : \mathbb{R}^n \to \mathbb{R}^n$, we can see:

$$\nabla \left(a \cdot (b_1, \dots, b_n)^T \right) = \left(\frac{\partial (a \cdot b_k)}{\partial x_i} \right)_{i,k \in \{1,\dots,n\}}$$

$$= \left(\frac{a}{\partial x_i} b_k + a \frac{\partial b_k}{\partial x_i} \right)_{i,k} = \nabla a b^T + a \cdot \nabla b$$

Computing the Hessian $\nabla^2 f(x)$

- $H_j = 0 \in \mathbb{R}^{n\times n}$ for $j \in \{1, \ldots, n\}$
- $$H_{i+n} = \begin{cases} 0 & \text{if } \omega_i \in \mathcal{C} \\ \omega_i'(f_{k_i})H_{k_i} + \omega_i''(f_{k_i})g_{k_i}g_{k_i}^T & \text{if } \omega_i \in \mathcal{U} \\ \partial_a\omega_i(f_{k_i}, f_{l_i})H_{k_i} + \partial_b\omega_i(f_{k_i}, f_{l_i})H_{l_i} & \\ +(g_{k_i}, g_{l_i})\nabla^2\omega_i(f_{k_i}, f_{l_i})(g_{k_i}, g_{l_i})^T & \text{if } \omega_i \in \mathcal{B} \end{cases}$$

Then we can easily see that:

$$\begin{aligned} \nabla\left(\omega_i'(f_{k_i})g_{k_i}\right) &= \nabla(\omega_i'(f_{k_i}))g_{k_i}^T + \omega_i'(f_{k_i})\left(\nabla g_{k_i}\right) \\ &= \omega_i'(f_{k_i})H_{k_i} + \omega_i''(f_{k_i})g_{k_i}g_{k_i}^T \end{aligned}$$

With ∇ being the *total Differential* Operator.

Computing the Hessian $\nabla^2 f(x)$

Not so easy to see is the case $\omega_i \in \mathcal{B}$:

$$
\begin{aligned}
&\nabla\left(\partial_a \omega_i(f_{k_i}, f_{l_i}) g_{k_i} + \partial_b \omega_i(f_{k_i}, f_{l_i}) g_{l_i}\right) \\
&= \nabla\left(\partial_a \omega_i(f_{k_i}, f_{l_i}) g_{k_i}^T\right) + \partial_a \omega_i(f_{k_i}, f_{l_i}) \cdot \nabla g_{k_i} \\
&+ \nabla\left(\partial_b \omega_i(f_{k_i}, f_{l_i}) g_{l_i}^T\right) + \partial_b \omega_i(f_{k_i}, f_{l_i}) \cdot \nabla g_{l_i} \\
\\
&= \partial_a \omega_i(f_{k_i}, f_{l_i}) \cdot H_{k_i} + \partial_b \omega_i(f_{k_i}, f_{l_i}) \cdot H_{l_i} \\
&+ \left(\partial_a^2 \omega_i(f_{k_i}, f_{l_i}) g_{k_i} + \partial_b \partial_a \omega_i(f_{k_i}, f_{l_i}) g_{l_i}\right) g_{k_i}^T \\
&+ \left(\partial_b^2 \omega_i(f_{k_i}, f_{l_i}) g_{l_i} + \partial_a \partial_b \omega_i(f_{k_i}, f_{l_i}) g_{k_i}\right) g_{l_i}^T
\end{aligned}
$$

Computing the Hessian $\nabla^2 f(x)$

$$
\begin{aligned}
&= \partial_a \omega_i(f_{k_i}, f_{l_i}) \cdot H_{k_i} + \partial_b \omega_i(f_{k_i}, f_{l_i}) \cdot H_{l_i} \\
&+ \left(\partial_a^2 \omega_i(f_{k_i}, f_{l_i}) g_{k_i} + \partial_b \partial_a \omega_i(f_{k_i}, f_{l_i}) g_{l_i}\right) g_{k_i}^T \\
&+ \left(\partial_b^2 \omega_i(f_{k_i}, f_{l_i}) g_{l_i} + \partial_a \partial_b \omega_i(f_{k_i}, f_{l_i}) g_{k_i}\right) g_{l_i}^T
\end{aligned}
$$

$$
\begin{aligned}
&= \partial_a \omega_i(f_{k_i}, f_{l_i}) \cdot H_{k_i} + \partial_b \omega_i(f_{k_i}, f_{l_i}) \cdot H_{l_i} \\
&+ (g_{k_i}, g_{l_i}) \begin{pmatrix} \partial_a^2 \omega_i(f_{k_i}, f_{l_i}) & \partial_a \partial_b \omega_i(f_{k_i}, f_{l_i}) \\ \partial_b \partial_a \omega_i(f_{k_i}, f_{l_i}) & \partial_b^2 \omega_i(f_{k_i}, f_{l_i}) \end{pmatrix} \begin{pmatrix} g_{k_i}^T \\ g_{l_i}^T \end{pmatrix}
\end{aligned}
$$

Computing the Hessian $\nabla^2 f(x)$

$$
\begin{aligned}
&= \partial_a \omega_i(f_{k_i}, f_{l_i}) \cdot H_{k_i} + \partial_b \omega_i(f_{k_i}, f_{l_i}) \cdot H_{l_i} \\
&+ (g_{k_i}, g_{l_i}) \begin{pmatrix} \partial_a^2 \omega_i(f_{k_i}, f_{l_i}) & \partial_a \partial_b \omega_i(f_{k_i}, f_{l_i}) \\ \partial_b \partial_a \omega_i(f_{k_i}, f_{l_i}) & \partial_b^2 \omega_i(f_{k_i}, f_{l_i}) \end{pmatrix} \begin{pmatrix} g_{k_i}^T \\ g_{l_i}^T \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
&= \partial_a \omega_i(f_{k_i}, f_{l_i}) \cdot H_{k_i} + \partial_b \omega_i(f_{k_i}, f_{l_i}) \cdot H_{l_i} \\
&+ \underbrace{(g_{k_i}, g_{l_i})}_{n \times 2} \underbrace{\nabla^2 \omega_i(f_{k_i}, f_{l_i})}_{2 \times 2} \underbrace{(g_{k_i}, g_{l_i})^T}_{2 \times n}
\end{aligned}
$$

Forward mode

Compute *sequentially*:

$$f_i,\ g_i,\ H_i,\ \text{for } i \in \{1, \ldots, n+m\}$$

and then we obtain:

- $f_{n+m} = f(x)$, the Function value,
- $g_{n+m} = \nabla f(x)$, the Gradient and
- $H_{n+m} = \nabla^2 f(x)$, the Hessian matrix.

This mode is called *forward mode*.

Reverse mode - layout

We define the following functions for $i \in \{1, \dots, m-1\}$

$$F_i : \mathbb{R}^{n+i-1} \to \mathbb{R}^{n+i}, F_i(y_1, \dots, y_{n+i-1}) = \begin{pmatrix} y_1 \\ \vdots \\ y_{n+i-1} \\ f_{i+n} \end{pmatrix}$$

And for $i = m$ we set:

$$F_m : \mathbb{R}^{n+i-1} \to \mathbb{R}, F_m\left(y_1, \dots, y_{n+m-1}\right) = f_{m+n}$$

Technically we can only define the F_i on open subsets of $\mathbb{R}^{n+i-1}$ on which the ω_i are defined, but here I left this out since it is obvious to see.

Reverse mode - layout

We set the intermediate state as $G_0 := x$ and

$$G_i^T := (f_1, \dots, f_{n+i})^T = F_i \circ F_{i-1} \circ \dots \circ F_1(x)$$

for $i \in \{1, \dots, m-1\}$. Thus we have the identity:

$$f(x) = f_{m+n} = F_m \circ F_{m-1} \circ \cdots \circ F_1(x)$$

Differentiating this identity w.r.t. x yields:

$$\nabla f(x)^T = DF_m(G_{m-1}) DF_{m-1}(G_{m-2}) \cdots DF_1(x) \quad (1)$$

Where DF denotes the Jacobian Matrix of F.

Reverse mode - layout

$$\nabla f(x)^T = DF_m(G_{m-1})DF_{m-1}(G_{m-2})\cdots DF_1(x) \quad (1)$$

Evaluating equation (1) from *right to left* corresponds to the forward mode of Auto-Diff.

Reverse mode - layout

$$\nabla f(x)^T = DF_m(G_{m-1})DF_{m-1}(G_{m-2}) \cdots DF_1(x) \quad (1)$$

In reverse mode, we want to evaluate (1) from *left to right*.

This obviously yields the *same* gradient $\nabla f(x)$.

Reverse mode - in detail

In detail we find:

$$DF_1(\overbrace{G_0}^{=x}) = \left(\begin{array}{c} I_n \\ \hline \frac{\partial f_{1+n}}{\partial x_1} \dots \frac{\partial f_{1+n}}{\partial x_n} \end{array} \right) \in \mathbb{R}^{(n+1)\times n}$$

With the last row being the gradient ∇f_{1+n}^T of f_{1+n}.

Reverse mode - in detail

This also works for $i \in \{1, \ldots, m-1\}$ and generalizes to:

$$DF_i(G_{i-1}) = \left(\begin{array}{c} I_{n+i-1} \\ \hline \frac{\partial f_{i+n}}{\partial f_1} \ldots \frac{\partial f_{i+n}}{\partial f_{n+i-1}} \end{array} \right) \in \mathbb{R}^{(n+i)\times(n+i-1)}$$

But now the last row is the gradient of f_{i+n} **w.r.t.** $(f_1, \ldots, f_{n+i-1})$ and not x !

Reverse mode - in detail

$$DF_i(G_{i-1}) = \begin{pmatrix} I_{n+i-1} \\ \hline \frac{\partial f_{i+n}}{\partial f_1} \cdots \frac{\partial f_{i+n}}{\partial f_{n+i-1}} \end{pmatrix} \in \mathbb{R}^{(n+i)\times(n+i-1)}$$

We now recognize that the f_{i+n} can only depend on at most 2 elements in $(f_1, \ldots, f_{n+i-1})$.

Thus the last row can only contain 2 non-zero elements, namely at indices k_i and l_i.

Reverse mode - in detail

$$DF_i(G_{i-1}) = \left(\begin{array}{ccc} & I_{n+i-1} & \\ \hline \frac{\partial f_{i+n}}{\partial f_1} & \cdots & \frac{\partial f_{i+n}}{\partial f_{n+i-1}} \end{array} \right) \in \mathbb{R}^{(n+i)\times(n+i-1)}$$

So we can write:

$$DF_i(G_{i-1}) = \left(\begin{array}{cc} \multicolumn{2}{c}{I_{n+i-1}} \\ \hline \kappa_i & \lambda_i \end{array} \right)$$

Reverse mode - in detail

So we can write:

$$DF_i(G_{i-1}) = \begin{pmatrix} & I_{n+i-1} & \\ \hline \kappa_i & & \lambda_i \end{pmatrix}$$

With κ_i and λ_i being the entries of the last row at indices k_i and l_i respectively, specifically:

$$\left(\frac{\partial f_{i+n}}{\partial f_1}, \ldots, \frac{\partial f_{i+n}}{\partial f_{n+i-1}}\right) = (\ldots, \kappa_i, \ldots, \lambda_i, \ldots)$$

The dots represent the zero-entries here.

Reverse mode - in detail

$$\left(\frac{\partial f_{i+n}}{\partial f_1}, \ldots, \frac{\partial f_{i+n}}{\partial f_{n+i-1}}\right) = (\ldots, \kappa_i, \ldots, \lambda_i, \ldots)$$

Notice that for $\omega_i \in \mathcal{U} \Rightarrow l_i = 0$ and in that case we only have one non-zero entry.

And if $\omega_i \in \mathcal{C}$ then the last row becomes entirely zero.

Reverse mode - in detail

We verify quickly:

$$\nabla f(x)^T = \underbrace{DF_m(G_{m-1})}_{(1\times n+m-1)} \underbrace{DF_{m-1}(G_{m-2})}_{(n+m-1\times n+m-2)} \cdots \underbrace{DF_1(x)}_{(n+1\times n)}$$

And we can see that this equation really results in a $(1 \times n)$ matrix, i.e. $\nabla f(x)^T$.

Reverse mode - in detail

The reverse mode is carried out in *m*-steps (like the forward mode). We compute recursively:

$$v^{(i)} := \begin{cases} DF^{(m)}(G_{m-1}) & \text{if } i = m \\ \\ v^{(i+1)} DF_i(G_{i-1}) & \text{if } i = m-1, \dots, 1 \end{cases}$$

Reverse mode - in detail

$$v^{(i)} := \begin{cases} DF^{(m)}(G_{m-1}) & \text{if } i = m \\ \\ v^{(i+1)} DF_i(G_{i-1}) & \text{if } i = m-1, \ldots, 1 \end{cases}$$

Then $v^{(1)}$ will be $\nabla f(x)^T$ according to the previous equation.

Reverse mode - in detail

$$v^{(i)} := \begin{cases} DF^{(m)}(G_{m-1}) & \text{if } i = m \\ v^{(i+1)} DF_i(G_{i-1}) & \text{if } i = m-1, \ldots, 1 \end{cases}$$

And we compute in the *reverse* direction:

$$v^{(m)} \to v^{(m-1)} \to \ldots \to v^{(2)} \to v^{(1)} = \nabla f(x)^T$$

Reverse mode - in detail

In detail we start with $v^{(m)} = (0, \ldots, \kappa_i, \ldots, \lambda_i, \ldots, 0)$:

And we inspect:

Reverse mode - in detail

In detail we start with $v^{(m)} = (0, \dots, \kappa_i, \dots, \lambda_i, \dots, 0)$:

And we inspect:

$$
\begin{aligned}
v^{(m-1)} &= v^{(m)} DF_{m-1}(G_{m-2}) \\
&= \begin{pmatrix} \kappa_i & \lambda_i \end{pmatrix}
\begin{array}{c}
\begin{pmatrix}
1 & 0 & \dots & \dots & 0 \\
0 & 1 & \ddots & \ddots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & \dots & \dots & 1
\end{pmatrix} \\
\begin{array}{ccccc} & \kappa_{i-1} & & \lambda_{i-1} & \end{array}
\end{array}
\end{aligned}
$$

Reverse mode - in detail

$$
\begin{aligned}
v^{(m-1)} &= v^{(m)} DF_{m-1}(G_{m-2}) \\
&= \begin{pmatrix} \kappa_i & \lambda_i \end{pmatrix}
\begin{array}{c}
\begin{pmatrix}
1 & 0 & \dots & \dots & 0 \\
0 & 1 & \ddots & \ddots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & \dots & \dots & 1
\end{pmatrix} \\
\begin{array}{ccccc} & \kappa_{i-1} & & \lambda_{i-1} & \end{array}
\end{array} \\
&= \left(v_1^{(m)}, v_2^{(m)}, \dots v_{n+m-2}^{(m)}, v_{n+m-1}^{(m)}\right) \\
&+ \left(0, \dots, \kappa_{m-1} v_{n+m}^{(m)}, 0, \dots, 0, \kappa_{m-1} v_{n+m}^{(m)}, \dots, 0\right)
\end{aligned}
$$

Reverse mode - in detail

$$
\begin{aligned}
&= \left(v_1^{(m)}, v_2^{(m)}, \ldots v_{n+m-2}^{(m)}, v_{n+m-1}^{(m)}\right) \\
&+ \left(0, \ldots, \kappa_{m-1} v_{n+m}^{(m)}, 0, \ldots, 0, \kappa_{m-1} v_{n+m}^{(m)}, \ldots, 0\right)
\end{aligned}
$$

To compute $v^{(i-1)}$ from $v^{(i)}$, we add $\kappa_{i-1} v_{n+i}^{(i)}$ to the k_i-th component and $\lambda_{i-1} v_{n+i}^{(i)}$ to the l_i-th component (if $k_i \neq 0$ and / or $l_i \neq 0$).

And of course delete the last component of $v^{(i)}$.

Reverse mode - in detail

Now remember that κ_i and λ_i are always entries of the gradients in g_i (the gradient vector).

Reverse mode - in detail

Now remember that κ_i and λ_i are always entries of the gradients in g_i (the gradient vector).

Compute $v^{(i)}$ efficiently by decomposing g_i and thus applying smart multiplication rules.

Applications - Optimization Problems

- Efficiently compute gradients *and Hessians*
- No need to *symbolically* calculate derivatives, especially for complex functions
- Auto-Diff achieves high numerical accuracy, better than numerical methods like finite differences

Applications - Neural Networks

General layout: Learn parameterized mapping $f_\theta : \mathcal{X} \to \mathcal{Y}$ from dataset $(x_i, y_i)_{i \in \{1,\dots,n\}}$ where: $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$.

1. Initialize neural network with random parameters θ

Applications - Neural Networks

General layout: Learn parameterized mapping $f_\theta : \mathcal{X} \to \mathcal{Y}$ from dataset $(x_i, y_i)_{i \in \{1,\dots,n\}}$ where: $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$.

1. Initialize neural network with random parameters θ
2. For samples from a dataset (x_i, y_i), calculate $f_\theta(x_i) = \hat{y}_i$ and the *gradient* of specified *loss function* L w.r.t. θ:

$$\nabla_\theta L(\hat{y}_i, y_i)$$

Applications - Neural Networks

General layout: Learn parameterized mapping $f_\theta : \mathcal{X} \to \mathcal{Y}$ from dataset $(x_i, y_i)_{i \in \{1,\dots,n\}}$ where: $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$.

1. Initialize neural network with random parameters θ
2. For samples from a dataset (x_i, y_i), calculate $f_\theta(x_i) = \hat{y}_i$ and the *gradient* of specified *loss function* L w.r.t. θ:

$$\nabla_\theta L(\hat{y}_i, y_i)$$

3. Update the parameters θ using the computed *gradient*

Applications - Neural Networks

General layout: Learn parameterized mapping $f_\theta : \mathcal{X} \to \mathcal{Y}$ from dataset $(x_i, y_i)_{i\in\{1,\dots,n\}}$ where: $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$.

$\to$ Auto-Diff computes mentioned gradient *at the same time* when computing $\hat{y}_i$ (the network outputs)

Literature used

- "Automatic Differentiation: A Structure-Exploiting Forward Mode with Almost Optimal Complexity for Kantorovic Trees" - Michael Ulbrich and Stefan Ulbrich January 1996

Thank you for your attention!