CS107 / AC207

SYSTEMS DEVELOPMENT FOR COMPUTATIONAL SCIENCE

LECTURE 10

Thursday, October 7th 2021

Fabian Wermelinger
Harvard University

RECAP OF LAST TIME

- Towards automatic differentiation
  - The Jacobian and Newton's method (root-finding)
  - Numerical computation of derivatives
- Finish Newton's method with exact and approximate Jacobian representations (catch up)

OUTLINE

Automatic Differentiation: ***Forward Mode*** (basics)

- Evaluation trace
- The computational graph
- Computing derivatives of one variable using the forward mode
- Computing derivatives in higher dimensions using the forward mode

Beyond the basics:

- The Jacobian in forward mode
- What the forward mode actually computes
- Implementation approaches

INTRODUCTION AND MOTIVATION

References for automatic differentiation:

- P. H.W. Hoffmann, *A Hitchhiker's Guide to Automatic Differentiation*, Springer 2015, doi:10.1007/s11075-015-0067-6 (You can access this paper through the Harvard network.)
- Griewank, A. and Walther, A., *Evaluating derivatives: principles and techniques of algorithmic differentiation*, SIAM 2008, Vol. 105
- Nocedal, J. and Wright, S., *Numerical Optimization*, Springer 2006, 2nd Edition

INTRODUCTION AND MOTIVATION

Differentiation is one of the most important operations in science.

- Finding extrema of functions and determining zeros of functions are central to optimization.
- Linearization of non-linear equations requires a prediction for a change in a small neighborhood which involves derivatives.
- *Numerically* solving differential equations forms a cornerstone of modern science and engineering and is intimately linked with predictive science.

THE BASIC IDEAS OF AUTOMATIC DIFFERENTIATION

- In the introduction, we motivated the need for computational techniques to compute derivatives.
- We have discussed the computation of J with symbolic math which is accurate but may not always be applicable depending on $f(x)$ or may be too costly to evaluate.
- Numerical computation of J may be an alternative method at the cost of accuracy reduction and possible stability issues.
- Automatic differentiation (AD) overcomes both of these deficiencies. It is
  - less costly than symbolic differentiation
  - evaluates derivatives to machine precision
- There are *two* modes of AD: ***forward*** and ***reverse***. The back-propagation algorithm in machine learning is a special case of the reverse AD mode.

REVIEW OF THE CHAIN RULE

At the heart of AD is the ***chain rule*** that you know from Calculus.

REVIEW OF THE CHAIN RULE

Suppose we have a function $h(u(t))$ and we want to compute the derivative of h with respect to t. This derivative is given by

$$\frac{dh}{dt} = \frac{\partial h}{\partial u}\frac{du}{dt}$$

Example: $h(u(t)) = \sin(4t)$ and $u(t) = 4t$

$$\frac{\partial h}{\partial u} = \cos(u), \quad \frac{du}{dt} = 4 \quad \Rightarrow \quad \frac{dh}{dt} = 4\cos(4t)$$

REVIEW OF THE CHAIN RULE

The *total change* of h is given by the sum of the partial changes in each coordinate direction.

Suppose h has another coordinate $v(t)$ so that we have $h(u(t), v(t))$. Once again, we want to compute the derivative of h with respect to t. Applying the chain rule in this case gives

$$\frac{dh}{dt} = \frac{\partial h}{\partial u}\frac{du}{dt} + \frac{\partial h}{\partial v}\frac{dv}{dt}$$

REVIEW OF THE CHAIN RULE

$$\frac{dh}{dt} = \frac{\partial h}{\partial u}\frac{du}{dt} + \frac{\partial h}{\partial v}\frac{dv}{dt}$$

Examples:

$$h(u(t), v(t)) = u + v \quad \Rightarrow \quad \frac{dh}{dt} = \frac{du}{dt} + \frac{dv}{dt}$$

$$h(u(t), v(t)) = uv \quad \Rightarrow \quad \frac{dh}{dt} = v\frac{du}{dt} + u\frac{dv}{dt}$$

$$h(u(t), v(t)) = \sin(uv) \quad \Rightarrow \quad \frac{dh}{dt} = v\cos(uv)\frac{du}{dt} + u\cos(uv)\frac{dv}{dt}$$

REVIEW OF THE CHAIN RULE

The gradient operator ∇:

In vector calculus, the gradient describes the fastest *increase* of a scalar function $h(x)$ along a certain spatial direction given by coordinates $x \in \mathbb{R}^m$. In our 3D world $m = 3$ but in general the coordinate x is m-dimensional. In 3D with coordinates $x = [x_1, x_2, x_3]^\intercal$, the gradient *operator* is given by

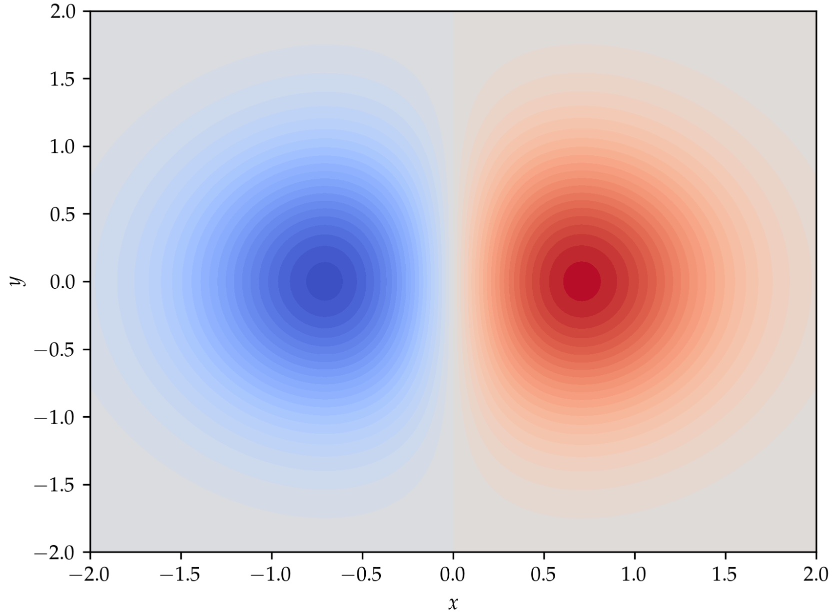
$$\nabla = \left[\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \frac{\partial}{\partial x_3}\right]^\intercal.$$

REVIEW OF THE CHAIN RULE

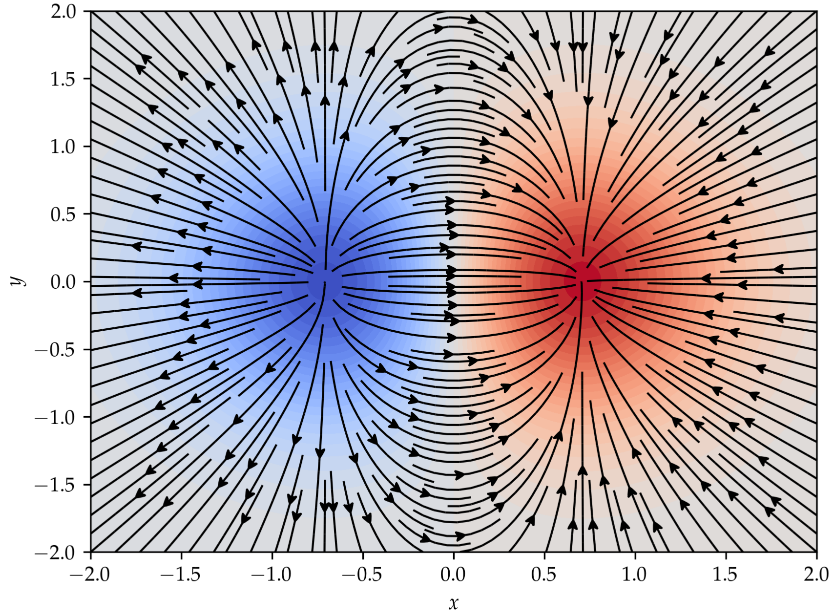
The gradient operator ∇:

Think of h as the *temperature* field T, then the *temperature gradient* ∇T describes the fastest *increase* of temperature T in a *certain direction*. Therefore, the temperature gradient is a *vector field*.

Temperature field T



Temperature gradient ∇T



REVIEW OF THE CHAIN RULE

The gradient operator ∇ (back to chain rule):

What happens if we replace the parameter $t \in \mathbb{R}$ from before with new coordinates $x \in \mathbb{R}^m$? We now want to compute the *gradient* of h with respect to x. We write $h(u(x), v(x))$ and we replace the d/dt operator from before with the gradient ∇:

$$\nabla_x h = \frac{\partial h}{\partial u} \nabla u + \frac{\partial h}{\partial v} \nabla v,$$

where we emphasize on the left side that the gradient is with respect to x. We do not write this on the right hand side because of $u = u(x)$ and $v = v(x)$ it is clear that the only possible gradient is with respect to x.

REVIEW OF THE CHAIN RULE

The gradient operator ∇ (back to chain rule):

$$\nabla_x h = \frac{\partial h}{\partial u}\nabla u + \frac{\partial h}{\partial v}\nabla v$$

The chain rule still holds, all we did is *replace the single coordinate t with an m-dimensional vector of coordinates x*. This required us to replace the differential operator d/dt with the differential vector operator ∇.

REVIEW OF THE CHAIN RULE

The gradient operator ∇ (back to chain rule):

$$\nabla_x h = \frac{\partial h}{\partial u}\nabla u + \frac{\partial h}{\partial v}\nabla v$$

Example:

Let $x = [x_1, x_2]^\intercal \in \mathbb{R}^2$, $u = u(x) = x_1 x_2$ and $v = v(x) = x_1 + x_2$.
Our function is given by $h(u, v) = \sin(u) - \cos(v)$

$$\nabla u = \begin{bmatrix} x_2 \\ x_1 \end{bmatrix}, \nabla v = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \Rightarrow \nabla_x h = \cos(x_1 x_2)\begin{bmatrix} x_2 \\ x_1 \end{bmatrix} + \sin(x_1 + x_2)\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

REVIEW OF THE CHAIN RULE

The (almost) general chain rule:

Let us now further generalize to not only $u = u(x)$ and $v = v(x)$ but many functions $y(x) = [y_1(x), \ldots, y_n(x)]^\intercal$ where all y_i take arguments $x \in \mathbb{R}^m$. Now $h = h(y(x))$ is a *scalar* function (therefore "almost" general chain rule) of possibly n other functions y_i, each themselves a function of m variables. The gradient of h is now given by:

$$\nabla_x h = \sum_{i=1}^{n} \frac{\partial h}{\partial y_i} \nabla y_i(x)$$

This is again the chain rule with n partial terms.

Relate to the example in the previous slide: $m = 2$ and $n = 2$ with $y_1 = u = x_1 x_2$ and $y_2 = v = x_1 + x_2$.

REVIEW OF THE CHAIN RULE

Spend 10 minutes with your neighbors:

- Make sure you feel comfortable with this notation.
- Help each other refresh on the ideas.
- Don't be scared of the general notation, the math behind simply is the chain rule.
- We just applied it assuming our function h depends on many other functions y_i which in turn are functions of many coordinates x_k.

EVALUATION (FORWARD) TRACE OF A FUNCTION

After the chain rule discussion above, let us apply the notation introduced and look at the evaluation trace of a scalar function $f(x)$ with a single argument $x \in \mathbb{R}$ ($m = 1$). Consider again the same function from the previous lecture:

$$f(x) = x - \exp\left(-2\left(\sin(4x)\right)^2\right).$$

We would like to evaluate the function at an arbitrary point x_1. Let us define $x_1 = \frac{\pi}{16}$.

EVALUATION (FORWARD) TRACE OF A FUNCTION

The correct evaluation of $f(x_1)$ involves a *partial ordering* of the operations associated with the function f.

For example: before we can evaluate $\sin(4x)$ we must evaluate the *intermediate* result $4x$ and before we can evaluate the exponential function we must evaluate the intermediate result $-2\left(\sin(4x)\right)^2$.

The evaluation trace introduces *intermediate results v_j for $j = 1, 2, \ldots$ of elementary binary operations like multiplying two numbers together or unary operations like computing* $\sin(v_j)$.

EVALUATION (FORWARD) TRACE OF A FUNCTION

A word on notation: the coordinates $x = [x_1, \dots, x_m]^\intercal$ that is $x \in \mathbb{R}^m$ are called *independent* variables, whereas the *intermediate results* v_j are *dependent* variables, they depend on x. We further *define the independent variables* as $v_{k-m} = x_k$ for $k = 1, 2, \dots, m$ in the following evaluation trace.

Recall: $f(x) = x - \exp\left(-2\left(\sin(4x)\right)^2\right)$ and we are interested in the value of $f(x_1 = \frac{\pi}{16})$:

EVALUATION (FORWARD) TRACE OF A FUNCTION

Recall: $f(x) = x - \exp\left(-2\left(\sin(4x)\right)^2\right)$ and we are interested in the value of $f(x_1 = \frac{\pi}{16})$:

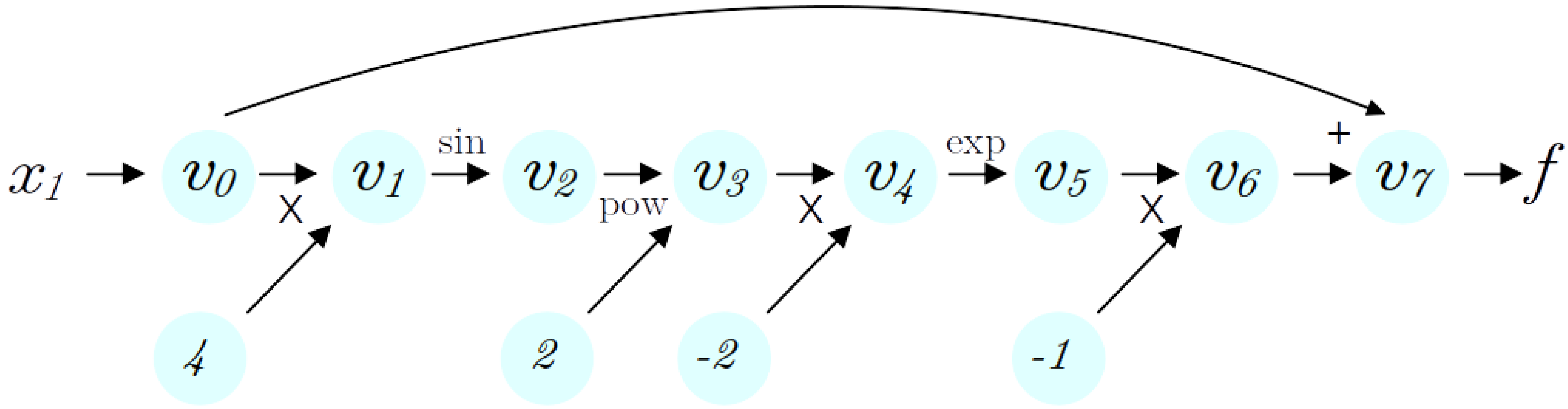
Intermediate	Elementary Operation	Numerical value
$v_0 = x_1$	$\frac{\pi}{16}$	1.963495e-01
v_1	$4v_0$	7.853982e-01
v_2	$\sin(v_1)$	7.071068e-01
v_3	v_2^2	5.000000e-01
v_4	$-2v_3$	-1.000000e+00
v_5	$\exp(v_4)$	3.678794e-01
v_6	$-v_5$	-3.678794e-01
$v_7 = f(x_1)$	$v_0 + v_6$	-1.715299e-01

Input variables (*independent* variables)

Intermediate variables (*dependent* variables, $v_j = v_j(x)$)

COMPUTATIONAL (FORWARD) GRAPH

We can think of each intermediate result v_j as a *node* in a *graph*. By doing so, we can get a visual interpretation of the partial ordering of *elementary operations* in $f(x) = x - \exp\left(-2\left(\sin(4x)\right)^2\right)$:



COMPUTATIONAL (FORWARD) GRAPH

The first ***key observation*** is that we worked *from the inside out* when developing the forward evaluation trace. We started from the value we want to evaluate $x_1 = \frac{\pi}{16}$ and built out to the actual function value $f(x_1)$. The second ***key observation*** is that in each evaluation step, we only carried out *elementary operations* between intermediate results v_j.

Later when we look at the *reverse mode* we will observe that it goes in the *opposite* direction.

COMPUTING THE DERIVATIVE AS WE GO ALONG

We are half-way through the forward mode of automatic differentiation:

- We have identified a partial ordering of elementary operations when evaluating an arbitrary function f.
- By breaking down the problem into smaller parts, we have computed intermediate results v_j for $j = 1, 2, \ldots$ where each $v_j = v_j(x)$ evaluated at point $x = x_1$.
- We have associated each v_j to a *node in a graph* for a visualization of the partial ordering. (Try to think about that in terms of a *data structure* as well.)

COMPUTING THE DERIVATIVE AS WE GO ALONG

Let us now return to the gradient ∇:

In the forward mode of automatic differentiation, we evaluate and carry forward a *directional derivative* of each intermediate variable v_j in a given direction $p \in \mathbb{R}^m$, *simultaneously* with the evaluation of v_j itself. (The latter is what we just did above.)

What does "direction" mean:

- Recall the linearization of the Euler equations (Lecture 9): the ***direction*** was the one that gave the *best linear approximation*.
- The flow rate through a surface is given by the flow velocity *normal to the surface* times the surface area. To get the normal velocity, we have to ***project*** it in the ***direction*** of the surface normal vector.
- In the forward AD mode: the ***direction*** is the one of a particular derivative we are interested in.

COMPUTING THE DERIVATIVE AS WE GO ALONG

Let us now return to the gradient ∇:

In the forward mode of automatic differentiation, we evaluate and carry forward a *directional derivative* of each intermediate variable v_j in a given direction $p \in \mathbb{R}^m$, *simultaneously* with the evaluation of v_j itself. (The latter is what we just did above.)

Therefore, let us *define* the gradient operator in a slightly different way than we did before. Here we ***project*** the gradient from before in the ***direction*** of p:

$$D_p y_i \stackrel{\text{def}}{=} (\nabla y_i)^\intercal p = \sum_{j=1}^{m} \frac{\partial y_i}{\partial x_j} p_j .$$

COMPUTING THE DERIVATIVE AS WE GO ALONG

$$D_p y_i \stackrel{\text{def}}{=} (\nabla y_i)^\top p = \sum_{j=1}^{m} \frac{\partial y_i}{\partial x_j} p_j.$$

Is the quantity $D_p y_i$ a ***vector*** or a ***scalar***?

COMPUTING THE DERIVATIVE AS WE GO ALONG

$$D_p y_i \stackrel{\text{def}}{=} (\nabla y_i)^\intercal p = \sum_{j=1}^{m} \frac{\partial y_i}{\partial x_j} p_j.$$

We now return to our example function from before

$$f(x) = x - \exp\left(-2\left(\sin(4x)\right)^2\right),$$

evaluated at the point $x = x_1 = \frac{\pi}{16}$. Be reminded that $x \in \mathbb{R}$ so $m = 1$. *There is only **one** possible direction of interest. Therefore, the natural choice is* $p = 1$ and we simply find:

$$D_p y_i = \frac{\partial y_i}{\partial x_1}$$

COMPUTING THE DERIVATIVE AS WE GO ALONG

Obviously we are after $D_p y_i$ in the forward mode of AD.

The y_i in $D_p y_i$ is *just a function* that depends on x. Let us say these functions correspond to the variables:

$$y_i = v_{i-m} \quad \text{for } i = 1, 2, \ldots, n,$$

where n is the sum of *independent* variables (the number m) and *intermediate* variables. In the forward trace we computed above for our example function we have $n = 8$ and $y_1 = v_0$, $y_2 = v_1$ up to $y_8 = v_7$.

COMPUTING THE DERIVATIVE AS WE GO ALONG

Before we continue and recompute the forward trace including the derivatives of the intermediate variables v_j, lets pick a few arbitrary intermediate steps and see how the last missing ingredient enters the forward mode of AD, that is the *chain rule*.

COMPUTING THE DERIVATIVE AS WE GO ALONG

What is the value of ∇v_0?

Note: from now on we no longer write ∇_x to indicate that the differentials are with respect to the independent coordinates x. We always assume this is the case.

We know that $v_0 = x_1$. Furthermore, we are only interested in the direction $p = 1$. Applying the result from before we find:

$$D_p v_0 = (\nabla v_0)^\intercal p = \frac{\partial x_1}{\partial x_1} \cdot 1 = 1$$

COMPUTING THE DERIVATIVE AS WE GO ALONG

What is the value of ∇v_2?

We know that $v_2 = v_2(v_1) = \sin(v_1)$. Because all v_j are functions of the independent coordinates x, *we must apply the chain rule here*:

$$\nabla v_2 = \frac{\partial v_2}{\partial v_1} \nabla v_1 = \cos(v_1) \nabla v_1$$

Again, we are only interested in the direction $p = 1$. Applying the result from before we find:

$$D_p v_2 = (\nabla v_2)^\top p = \cos(v_1)(\nabla v_1)^\top p = \cos(v_1) D_p v_1$$

Observe: we can compute the derivative of v_j with knowledge of v_i and $D_p v_i$ for $i < j$.

COMPUTING THE DERIVATIVE AS WE GO ALONG

What is the value of ∇v_7?

We apply what we know: $v_7 = v_7(v_0, v_6) = v_0 + v_6$. Nothing new here ***except*** that we further know $v_7 = f(x_1)$ such that $\nabla v_7 = \nabla f$ and the directional derivative $D_p v_7 = D_p f$ is *exactly* the derivative we are after, *evaluated* at coordinate x_1:

$$\nabla v_7 = \frac{\partial v_7}{\partial v_0} \nabla v_0 + \frac{\partial v_7}{\partial v_6} \nabla v_6 .$$

Projection in direction of p yields:

$$D_p v_7 = D_p v_0 + D_p v_6 ,$$

where $\partial(v_0 + v_6)/\partial v_0 = 1$ and $\partial(v_0 + v_6)/\partial v_6 = 1$.

COMPUTING THE DERIVATIVE AS WE GO ALONG

We now repeat the computation of the forward trace for our test function $f(x)$. What we did earlier is called the forward *primal* trace, we extend it this time with the forward *tangent* trace which corresponds to the derivatives of the intermediate variables.

In the forward mode of automatic differentiation, we evaluate and carry forward a *directional derivative* $D_p v_j$ of each intermediate variable v_j in a given direction $p \in \mathbb{R}^m$, *simultaneously* with the evaluation of v_j itself.

Recall: $f(x) = x - \exp\left(-2\left(\sin(4x)\right)^2\right)$ and we are interested in the value of $\left.\frac{\partial f}{\partial x}\right|_{x=x_1}$:

AUTOMATIC DIFFERENTIATION: FORWARD MODE

Recall: $f(x) = x - \exp\left(-2\left(\sin(4x)\right)^2\right)$ and we are interested in the value of $\left.\frac{\partial f}{\partial x}\right|_{x=x_1}$:

Forward primal trace	Forward tangent trace	Numerical value: v_j; $D_p v_j$
$v_0 = x_1 = \frac{\pi}{16}$	$D_p v_0 = 1$	1.963495e-01; 1.000000e+00
$v_1 = 4v_0$	$D_p v_1 = 4D_p v_0$	7.853982e-01; 4.000000e+00
$v_2 = \sin(v_1)$	$D_p v_2 = \cos(v_1) D_p v_1$	7.071068e-01; 2.828427e+00
$v_3 = v_2^2$	$D_p v_3 = 2v_2 D_p v_2$	5.000000e-01; 4.000000e+00
$v_4 = -2v_3$	$D_p v_4 = -2D_p v_3$	-1.000000e+00;-8.000000e+00
$v_5 = \exp(v_4)$	$D_p v_5 = \exp(v_4) D_p v_4$	3.678794e-01;-2.943036e+00
$v_6 = -v_5$	$D_p v_6 = -D_p v_5$	-3.678794e-01; 2.943036e+00
$v_7 = f(x_1) = v_0 + v_6$	$D_p v_7 = \left.\frac{\partial f}{\partial x}\right\|_{x=x_1} = D_p v_0 + D_p v_6$	-1.715299e-01; 3.943036e+00

Input variables (*independent* variables)

Intermediate variables (*dependent* variables, $v_j = v_j(x)$)

AUTOMATIC DIFFERENTIATION: FORWARD MODE

Recall: $f(x) = x - \exp\left(-2\left(\sin(4x)\right)^2\right)$ and we are interested in the value of $\left.\frac{\partial f}{\partial x}\right|_{x=x_1}$:

Forward primal trace	Forward tangent trace	Numerical value: v_j; $D_p v_j$
$v_0 = x_1 = \frac{\pi}{16}$	$D_p v_0 = 1$	1.963495e-01; 1.000000e+00
$v_7 = f(x_1) = v_0 + v_6$	$D_p v_7 = \left.\frac{\partial f}{\partial x}\right|_{x=x_1} = D_p v_0 + D_p v_6$	-1.715299e-01; 3.943036e+00

We have computed the derivative last time on paper and with `sympy`. You are encouraged to check that we indeed compute the correct result.

AUTOMATIC DIFFERENTIATION: FORWARD MODE

That is all there is to forward mode AD. The key observations are the following:

- We have broken down the evaluation of an arbitrary function $f(x)$ into smaller pieces, each only consists of *elementary* operations like addition, multiplication, division, subtraction, exponentiation, trigonometric functions and so on.
- Forward mode works from the inside out.
- We have computed a primal trace of intermediate variables v_j and a tangent trace of their directional derivatives $D_p v_j$ both *simultaneously* in the same step.
- Since we only work with elementary functions, we know their derivatives and computing $D_p v_j$ is a trivial task.

AUTOMATIC DIFFERENTIATION: FORWARD MODE

Some comments on implementation:

- The computational graph we studied earlier identifies the *nodes* associated to intermediate variables v_j. The evaluation of v_j depends on its *parents* in the graph. Node v_i is a *parent* of the *child* node v_j whenever there is a *directed* arc from i to j. This implies a (data) structure that you will need to work with in your AD library.
- There is ***no need*** to construct the computational graph, break down the problem into its partial ordering or identify intermediate variables *manually*. Automatic (or a better word is *algorithmic*) differentiation software can perform these tasks implicitly via the implemented algorithm and data structure.
- Once a child node is evaluated, its parent node(s) are no longer needed (if the parent has no more other children that must be evaluated) and can therefore be overwritten or discarded. ***There is no need to store the full graph of v_j and $D_p v_j$ pairs.*** This is a strength of the forward mode as the computational graph can become very large for non-trivial functions $f(x)$.

AUTOMATIC DIFFERENTIATION: FORWARD MODE

Another word on notation: in the literature you may come across the notation $\dot{v}_j$ to denote the directional derivative of v_j, instead of the notation $D_p v_j$ that we have used here. In physics, the "dot" notation refers to differentiation with respect to time, which can only advance in one direction. Our direction is given by the m-dimensional vector p for which the notation $D_p v_j$ seems more precise.
Read it as: *derivative of v_j in direction of p*.

Of course you are free to choose whichever notation you are most comfortable with.

FORWARD MODE AD: HIGHER DIMENSIONS

So far we have been looking at a scalar function $f(x)$ with a single argument $x \in \mathbb{R}$. In the following slides we extend our discussion to:

- Multivariate scalar function $f(x) : \mathbb{R}^m \mapsto \mathbb{R}$
- Multivariate vector function $f(x) : \mathbb{R}^m \mapsto \mathbb{R}^n$

The mathematics we covered up to here remains exactly the same, what changes is the number of inputs and outputs in the computational graph.

FORWARD MODE AD: HIGHER DIMENSIONS

We start by looking at the case $f(x) : \mathbb{R}^m \mapsto \mathbb{R}$, where $x \in \mathbb{R}^m$.

- We deal with more than one *input* $x = [x_1, x_2, \ldots, x_m]^\mathsf{T}$.
- This means we have m *independent* variables. If you recall the table for the primal and tangent traces, we have m gray rows instead of just one. Similarly, the computational graph will have m input nodes on the left side.
- The direction $p \in \mathbb{R}^m$ has m components too.

FORWARD MODE AD: HIGHER DIMENSIONS

More notation: the vector $p \in \mathbb{R}^m$ is called the *seed vector*. We have introduced it when we defined our directional derivative:

$$D_p y_i \stackrel{\text{def}}{=} (\nabla y_i)^\intercal p = \sum_{j=1}^{m} \frac{\partial y_i}{\partial x_j} p_j.$$

This definition is just a weighted sum (inner product) of derivatives with respect to the independent variables. The "direction" is given by the seed vector p.

The seed vector allows us to *cherry-pick a certain derivative of interest* (choose a "direction"). If we were interested in $\partial y_i / \partial x_1$ we would choose $p_1 = 1$ and $p_k = 0 \; \forall k \neq 1$. We can even choose a weighted combination of derivatives $\partial y_i / \partial x_j$ if we needed to.

We are free to choose the seed vector p.

FORWARD MODE AD: HIGHER DIMENSIONS

Example: 2-dimensional input ($m = 2$)

Consider the independent coordinates $x = [x_1, x_2]^\top$ with

$$f(x) = x_1 x_2.$$

It is easy to compute the gradient right away:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} x_2 \\ x_1 \end{bmatrix}.$$

FORWARD MODE AD: HIGHER DIMENSIONS

Example: 2-dimensional input ($m = 2$)

The primal trace consists of simply *one* intermediate variable

$$f(x) = v_1 = v_{-1} v_0 = x_1 x_2 \,.$$

The tangent trace requires the computation of $D_p v_1$, where now $p = [p_1, p_2]^\intercal$:

$$D_p v_1 = (\nabla v_1)^\intercal p = \frac{\partial v_1}{\partial x_1} p_1 + \frac{\partial v_1}{\partial x_2} p_2 = x_2 p_1 + x_1 p_2$$

- How do you choose p if you are interested in $\frac{\partial f}{\partial x_1}$? $p = [1, 0]^\intercal$
- How do you choose p if you are interested in $\frac{\partial f}{\partial x_2}$? $p = [0, 1]^\intercal$

FORWARD MODE AD: HIGHER DIMENSIONS

Example: 2-dimensional input ($m = 2$)

Consider now the function

$$f(x) = \sin(x_1 x_2).$$

The primal trace consists of *two* intermediate variables

$$\begin{aligned} v_1 &= v_{-1} v_0 = x_1 x_2 \\ f(x) = v_2 &= \sin(v_1) \end{aligned}$$

From the previous slide you know that:

$$D_p v_1 = x_2 p_1 + x_1 p_2$$

Spend ***10 minutes*** with your neighbors and go through the seed vector slides for our $m = 2$ example. ***Draw the computational graph of the problem. What is the value of*** $D_p v_2$***?***

FORWARD MODE AD: HIGHER DIMENSIONS

From this example we see that what the ***forward mode*** in AD really computes is:

$$\nabla f \cdot p$$

If the mapping is of the most general form $f(x) : \mathbb{R}^m \mapsto \mathbb{R}^n$, that is, f ***is a vector function***, then the product ∇f is an *outer product* that turns a vector into a rank-2 tensor (think of it as matrix that has a direction). The elements of that matrix are given by $\frac{\partial f_i}{\partial x_j}$ and we know from the previous lecture that this is called the ***Jacobian*** J.

In the general case, forward mode in AD computes the inner product of the Jacobian with the seed vector p

$$J \cdot p,$$

where $J \in \mathbb{R}^{n \times m}$ and $p \in \mathbb{R}^m$.

FORWARD MODE AD: HIGHER DIMENSIONS

In this last example we consider the mapping $f(x) : \mathbb{R}^2 \mapsto \mathbb{R}^2$, that is $m = 2$ and $n = 2$. The *vector valued function* is given by:

$$f(x) = \begin{bmatrix} x_1 x_2 + \sin(x_1) \\ x_1 + x_2 + \sin(x_1 x_2) \end{bmatrix},$$

where $x = [x_1, x_2]^\intercal$.

The first derivatives for this function are easy to compute:

$$\nabla f = J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} x_2 + \cos(x_1) & x_1 \\ 1 + x_2 \cos(x_1 x_2) & 1 + x_1 \cos(x_1 x_2) \end{bmatrix}$$

FORWARD MODE AD: HIGHER DIMENSIONS

In this last example we consider the mapping $f(x) : \mathbb{R}^2 \mapsto \mathbb{R}^2$, that is $m = 2$ and $n = 2$. The *vector valued function* is given by:

$$f(x) = \begin{bmatrix} x_1 x_2 + \sin(x_1) \\ x_1 + x_2 + \sin(x_1 x_2) \end{bmatrix},$$

where $x = [x_1, x_2]^\intercal$.

What is the computational graph for this problem?

FORWARD MODE AD: HIGHER DIMENSIONS

We want to compute the directional derivative $D_p v_5 = D_p f_1$, that is the *first component* of the vector function. By drawing the computational graph we should have found that $v_5 = v_1 + v_2$:

$$D_p v_5 = (\nabla v_5)^\intercal p = \underbrace{\left(\frac{\partial v_5}{\partial v_1} \nabla v_1 + \frac{\partial v_5}{\partial v_2} \nabla v_2 \right)}_{\text{chain rule}}^\intercal p = (\nabla v_1 + \nabla v_2)^\intercal p$$

$$= D_p v_1 + D_p v_2$$

FORWARD MODE AD: HIGHER DIMENSIONS

$$D_p v_5 = D_p v_1 + D_p v_2$$

We need $D_p v_1$ and $D_p v_2$. From the graph we know $v_1 = v_{-1} v_0$:

$$D_p v_1 = D_p(v_{-1} v_0) = \underbrace{v_0 D_p v_{-1} + v_{-1} D_p v_0}_{\text{product rule}}$$

but $v_{-1} = x_1$ and $v_0 = x_2$:

$$D_p v_{-1} = (\nabla v_{-1})^\intercal p = \begin{bmatrix} \frac{\partial x_1}{\partial x_1} \\ \frac{\partial x_1}{\partial x_2} \end{bmatrix}^\intercal p = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = p_1$$

If you do the same math for $D_p v_0$ you find:

$$D_p v_1 = x_2 p_1 + x_1 p_2$$

FORWARD MODE AD: HIGHER DIMENSIONS

$$D_p v_5 = D_p v_1 + D_p v_2$$

If you follow the same procedure for $D_p v_2$ where $v_2 = \sin(v_{-1})$ you find the following solution:

$$D_p v_5 = D_p f_1 = \left(x_2 + \cos(x_1)\right) p_1 + x_1 p_2$$

If we choose $p = [1, 0]^\intercal$ (the ***unit vector*** for coordinate x_1) then $D_p v_5 = \frac{\partial f_1}{\partial x_1}$, which is exactly the first element in the first row of the Jacobian J. If we choose $p = [0, 1]^\intercal$ then we get the second element of the first row. The elements of the second row are obtained by computing $D_p v_6$ in the same way.

Take-home message: we can form the full Jacobian by using m unit vectors (as seed vectors) where m is the number of independent variables.

RECAP

Automatic Differentiation: ***Forward Mode*** (basics)

- Evaluation trace
- The computational graph
- Computing derivatives of one variable using the forward mode
- Computing derivatives in higher dimensions using the forward mode

Beyond the basics:

- The Jacobian in forward mode
- What the forward mode actually computes
- Implementation approaches