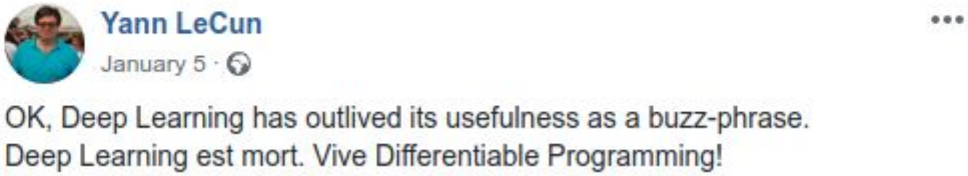
Automatic differentiation

Code & Cake

Fran Bartolić

fbartolic

Differentiable programming



Yann LeCun
January 5 ·

OK, Deep Learning has outlived its usefulness as a buzz-phrase.
Deep Learning est mort. Vive Differentiable Programming!

Three kinds of automated differentiation

- **Symbolic differentiation (e.g. Mathematica)**
  - Exact method of calculating derivatives by manipulating symbolic expressions
  - Memory intensive and very slow
- **Numerical differentiation (e.g. Finite differences)**
  - Easy to code, but subject to floating point errors, very slow in high dimensions

$$\frac{\partial}{x_i} f(x_1, \ldots x_N) \approx \frac{f(x_1, \ldots, x_i + h, \ldots, x_N) - f(x_1, \ldots, x_i - h, \ldots, x_N)}{2h}$$

- **Automatic differentiation (e.g. PyTorch, Tensorflow)**
  - Exact, speed comparable to analytic derivatives
  - Difficult to implement

What is Automatic Differentiation?

A Simple Automatic Derivative Evaluation Program

R. E. WENGERT
General Electric Company, Syracuse, New York*

A procedure for automatic evaluation of total/partial derivatives of arbitrary algebraic functions is presented. The technique permits computation of numerical values of derivatives without developing analytical expressions for the derivatives. The key to the method is the decomposition of the given function, by introduction of intermediate variables, into a series of elementary functional steps. A library of elementary function subroutines is provided for the automatic evaluation and differentiation of these new variables. The final step in this process produces the desired function's derivative.

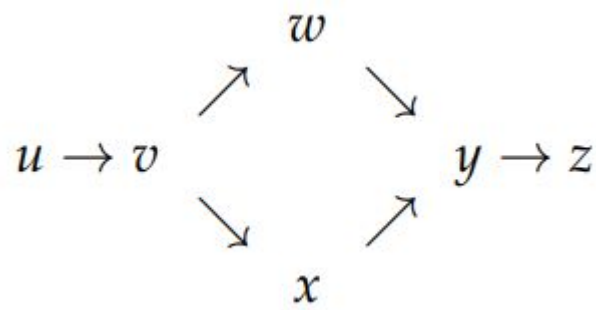
The main feature of this approach is its simplicity. It can be used as a quick-reaction tool where the derivation of analytical derivatives is laborious and also as a debugging tool for programs which contain derivatives.

What is Automatic Differentiation?

- A function written in a given programming language (e.g. Python, C++) is a composition of a finite number of elementary operations such as +, -, *, /, exp, sin, cos, etc.
- We know how to differentiate those elementary functions
- Therefore we can decompose an arbitrarily complicated function, differentiate the elementary parts and apply the chain rule to get exact derivatives of function outputs w.r. to inputs

Forward mode Automatic Differentiation

$$z = \exp(\sin(u^2)\log(u^2)) \quad \Longleftrightarrow$$

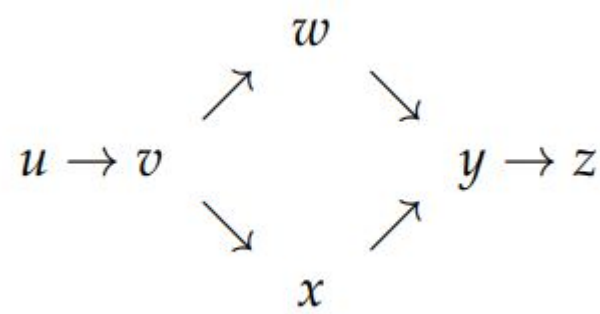


$$\dot{\theta} \equiv \frac{\partial \theta}{\partial u}, \quad \text{where } \theta \text{ is any intermediate quantity}$$

See Ian Murray's MLPR course notes for more details

Forward mode Automatic Differentiation

$$z = \exp(\sin(u^2)\log(u^2)) \quad \Longleftrightarrow$$



$$\dot{\theta} \equiv \frac{\partial \theta}{\partial u}, \quad \text{where } \theta \text{ is any intermediate quantity}$$

$$\dot{v} = \frac{\partial v}{\partial u}$$

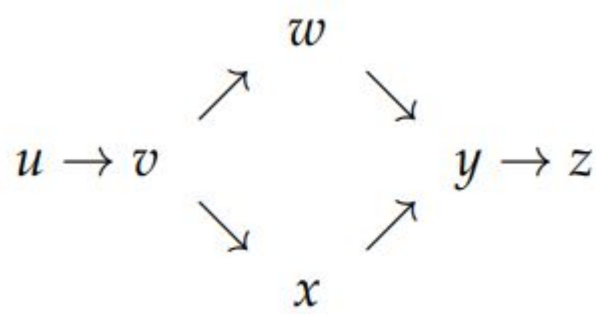
$$= 2u$$

See Ian Murray's MLPR course notes for more details

Forward mode Automatic Differentiation

$$z = \exp(\sin(u^2)\log(u^2))$$

$\Longleftrightarrow$



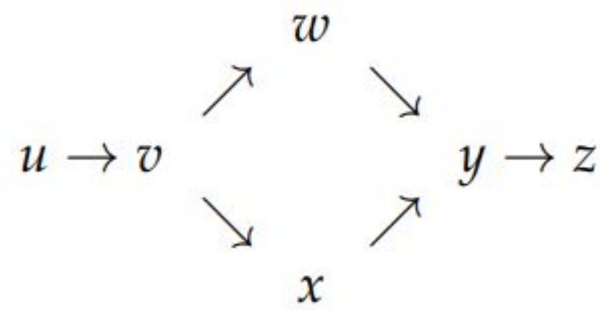
$$\dot{\theta} \equiv \frac{\partial \theta}{\partial u}, \quad \text{where } \theta \text{ is any intermediate quantity}$$

$$\begin{aligned} \dot{v} &= \frac{\partial v}{\partial u} \\ &= 2u \end{aligned} \qquad \begin{aligned} \dot{w} &= \frac{\partial w}{\partial u} \\ &= \frac{\partial w}{\partial v}\frac{\partial v}{\partial u} \\ &= \cos(v)\dot{v} \end{aligned}$$

See Ian Murray’s MLPR course notes for more details

Forward mode Automatic Differentiation

$$z = \exp(\sin(u^2)\log(u^2)) \iff$$



$$\dot{\theta} \equiv \frac{\partial \theta}{\partial u}, \quad \text{where } \theta \text{ is any intermediate quantity}$$

$$\dot{v} = \frac{\partial v}{\partial u} = 2u \qquad \dot{w} = \frac{\partial w}{\partial u} = \frac{\partial w}{\partial v}\frac{\partial v}{\partial u} = \cos(v)\dot{v} \qquad \dot{x} = \frac{\partial x}{\partial u} = \frac{\partial x}{\partial v}\frac{\partial v}{\partial u} = (1/v)\dot{v}$$

See Ian Murray’s MLPR course notes for more details

Forward mode Automatic Differentiation

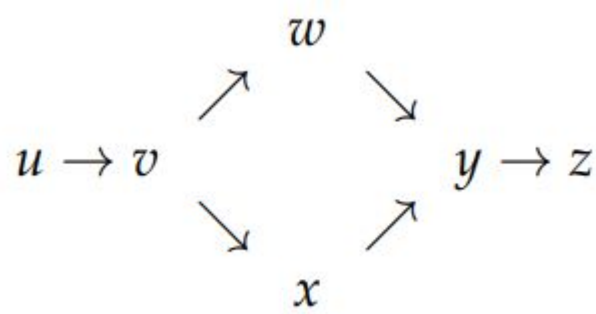
$$z = \exp(\sin(u^2)\log(u^2)) \quad \Longleftrightarrow \quad u \to v \begin{array}{ccc} \nearrow & w & \searrow \\ & & \\ \searrow & x & \nearrow \end{array} y \to z$$

$$\dot{\theta} \equiv \frac{\partial \theta}{\partial u}, \quad \text{where } \theta \text{ is any intermediate quantity}$$

$$\begin{aligned} \dot{v} &= \frac{\partial v}{\partial u} \\ &= 2u \end{aligned} \qquad \begin{aligned} \dot{w} &= \frac{\partial w}{\partial u} \\ &= \frac{\partial w}{\partial v}\frac{\partial v}{\partial u} \\ &= \cos(v)\dot{v} \end{aligned} \qquad \begin{aligned} \dot{x} &= \frac{\partial x}{\partial u} \\ &= \frac{\partial x}{\partial v}\frac{\partial v}{\partial u} \\ &= (1/v)\dot{v} \end{aligned} \qquad \begin{aligned} \dot{y} &= \frac{\partial y}{\partial u} \\ &= \frac{\partial y}{\partial w}\frac{\partial w}{\partial u} + \frac{\partial y}{\partial x}\frac{\partial x}{\partial u} \\ &= x\dot{w} + w\dot{x} \end{aligned}$$

See Ian Murray’s MLPR course notes for more details

$$z = \exp(\sin(u^2)\log(u^2)) \quad \Longleftrightarrow$$



$$\dot{\theta} \equiv \frac{\partial \theta}{\partial u}, \quad \text{where } \theta \text{ is any intermediate quantity}$$

$$\begin{aligned} \dot{v} &= \frac{\partial v}{\partial u} \\ &= 2u \end{aligned}$$

$$\begin{aligned} \dot{w} &= \frac{\partial w}{\partial u} \\ &= \frac{\partial w}{\partial v}\frac{\partial v}{\partial u} \\ &= \cos(v)\dot{v} \end{aligned}$$

$$\begin{aligned} \dot{x} &= \frac{\partial x}{\partial u} \\ &= \frac{\partial x}{\partial v}\frac{\partial v}{\partial u} \\ &= (1/v)\dot{v} \end{aligned}$$

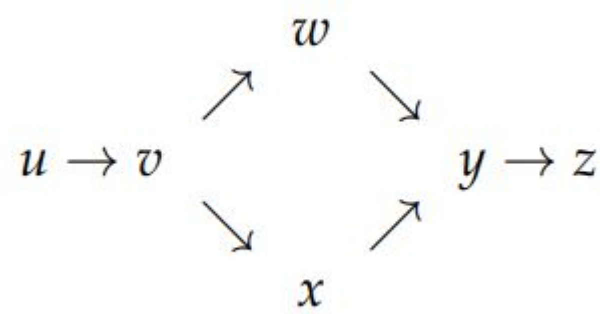
$$\begin{aligned} \dot{y} &= \frac{\partial y}{\partial u} \\ &= \frac{\partial y}{\partial w}\frac{\partial w}{\partial u} + \frac{\partial y}{\partial x}\frac{\partial x}{\partial u} \\ &= x\dot{w} + w\dot{x} \end{aligned}$$

$$\begin{aligned} \dot{z} &= \frac{\partial z}{\partial u} \\ &= \frac{\partial z}{\partial y}\frac{\partial y}{\partial u} \\ &= z\dot{y} \end{aligned}$$

See Ian Murray's MLPR course notes for more details

Reverse mode Automatic differentiation (Backpropagation)

$$z = \exp(\sin(u^2)\log(u^2)) \iff$$



$$\bar{\theta} \equiv \frac{\partial z}{\partial \theta}, \quad \text{where } \theta \text{ is any intermediate quantity}$$

See Ian Murray's MLPR course notes for more details

Reverse mode Automatic differentiation (Backpropagation)

$$z = \exp(\sin(u^2)\log(u^2)) \quad \Longleftrightarrow \quad \begin{array}{ccccc} & & w & & \\ & \nearrow & & \searrow & \\ u \rightarrow v & & & & y \rightarrow z \\ & \searrow & & \nearrow & \\ & & x & & \end{array}$$

$$\bar{\theta} \equiv \frac{\partial z}{\partial \theta}, \quad \text{where } \theta \text{ is any intermediate quantity}$$

$$\begin{aligned} \bar{y} &= \frac{\partial z}{\partial y} \\ &= \exp(y) \\ &= z \end{aligned}$$

See Ian Murray's MLPR course notes for more details

Reverse mode Automatic differentiation (Backpropagation)

$$z = \exp(\sin(u^2)\log(u^2)) \quad \Longleftrightarrow \quad \begin{array}{ccccc} & & w & & \\ & \nearrow & & \searrow & \\ u \rightarrow v & & & & y \rightarrow z \\ & \searrow & & \nearrow & \\ & & x & & \end{array}$$

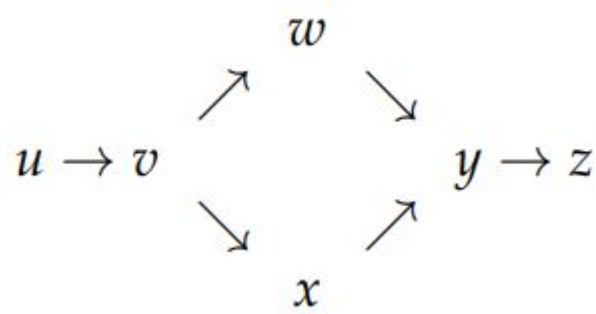
$$\bar{\theta} \equiv \frac{\partial z}{\partial \theta}, \quad \text{where } \theta \text{ is any intermediate quantity}$$

$$\begin{aligned} \bar{x} &= \frac{\partial z}{\partial x} & \bar{y} &= \frac{\partial z}{\partial y} \\ &= \frac{\partial z}{\partial y}\frac{\partial y}{\partial x} & &= \exp(y) \\ &= \bar{y}w & &= z \end{aligned}$$

See Ian Murray's MLPR course notes for more details

Reverse mode Automatic differentiation (Backpropagation)

$$z = \exp(\sin(u^2)\log(u^2)) \iff$$



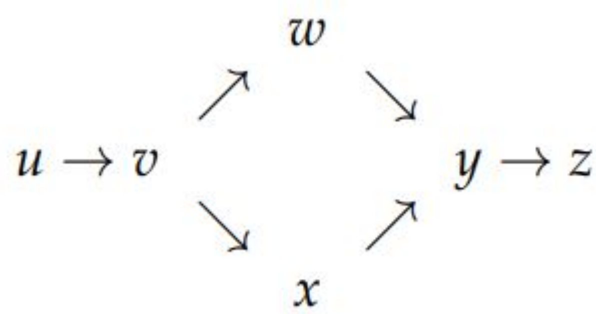
$$\bar{\theta} \equiv \frac{\partial z}{\partial \theta}, \quad \text{where } \theta \text{ is any intermediate quantity}$$

$$\bar{w} = \frac{\partial z}{\partial w} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial w} = \bar{y}x \qquad \bar{x} = \frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x} = \bar{y}w \qquad \bar{y} = \frac{\partial z}{\partial y} = \exp(y) = z$$

See Ian Murray’s MLPR course notes for more details

Reverse mode Automatic differentiation (Backpropagation)

$$z = \exp(\sin(u^2)\log(u^2)) \quad \Longleftrightarrow$$



$$\bar{\theta} \equiv \frac{\partial z}{\partial \theta}, \quad \text{where } \theta \text{ is any intermediate quantity}$$

$$\begin{aligned}
\bar{v} &= \frac{\partial z}{\partial v} \\
&= \frac{\partial z}{\partial w}\frac{\partial w}{\partial v} + \frac{\partial z}{\partial x}\frac{\partial x}{\partial v} \\
&= \bar{w}\cos(v) + \bar{x}(1/v)
\end{aligned}
\qquad
\begin{aligned}
\bar{w} &= \frac{\partial z}{\partial w} \\
&= \frac{\partial z}{\partial y}\frac{\partial y}{\partial w} \\
&= \bar{y}x
\end{aligned}
\qquad
\begin{aligned}
\bar{x} &= \frac{\partial z}{\partial x} \\
&= \frac{\partial z}{\partial y}\frac{\partial y}{\partial x} \\
&= \bar{y}w
\end{aligned}
\qquad
\begin{aligned}
\bar{y} &= \frac{\partial z}{\partial y} \\
&= \exp(y) \\
&= z
\end{aligned}$$

See Ian Murray's MLPR course notes for more details

Reverse mode Automatic differentiation (Backpropagation)

$$z = \exp(\sin(u^2)\log(u^2)) \quad \Longleftrightarrow \quad \begin{array}{ccccc} & & w & & \\ & \nearrow & & \searrow & \\ u \to v & & & & y \to z \\ & \searrow & & \nearrow & \\ & & x & & \end{array}$$

$$\bar{\theta} \equiv \frac{\partial z}{\partial \theta}, \quad \text{where } \theta \text{ is any intermediate quantity}$$

$$\begin{aligned} \bar{u} &= \frac{\partial z}{\partial u} & \bar{v} &= \frac{\partial z}{\partial v} & \bar{w} &= \frac{\partial z}{\partial w} & \bar{x} &= \frac{\partial z}{\partial x} & \bar{y} &= \frac{\partial z}{\partial y} \\ &= \frac{\partial z}{\partial v}\frac{\partial v}{\partial u} & &= \frac{\partial z}{\partial w}\frac{\partial w}{\partial v} + \frac{\partial z}{\partial x}\frac{\partial x}{\partial v} & &= \frac{\partial z}{\partial y}\frac{\partial y}{\partial w} & &= \frac{\partial z}{\partial y}\frac{\partial y}{\partial x} & &= \exp(y) \\ &= \bar{v}(2u) & &= \bar{w}\cos(v) + \bar{x}(1/v) & &= \bar{y}x & &= \bar{y}w & &= z \end{aligned}$$

See Ian Murray's MLPR course notes for more details

Forward vs. reverse mode Automatic Differentiation

- **Forward mode automatic differentiation**
  - We accumulate the derivatives in a *forward pass* through the graph, can do this parallel with function evaluation, differentiate w.r. to input
  - Don’t need to store the whole graph in memory
  - If we have multiple inputs we need to do a forward pass for each input
- **Reverse mode automatic differentiation (backpropagation)**
  - We accumulate the derivatives in a *reverse pass* through the graph, differentiate intermediate quantity w.r. to output
  - The computation can no longer be done in parallel with the function, need to store whole graph in memory
  - One reverse-mode pass gives us derivatives w.r. to *all inputs*!

Autodiff in Python: Autograd module

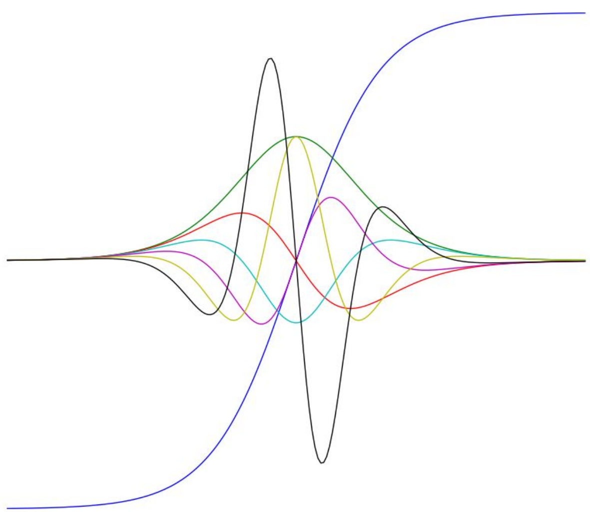
- Automatic differentiation in Python available via Autograd module, can be installed with `pip install autograd`
- Autograd can automatically differentiate most Python and Numpy code, it handles loops, `if` statements and recursion and closures
- Can do both forward mode autodiff and backpropagation
- Can handle higer-order derivatives

Autodiff in Python: Autograd module

```
>>> import autograd.numpy as np  # Thinly-wrapped numpy
>>> from autograd import grad    # The only autograd function you may ever need
>>>
>>> def tanh(x):                 # Define a function
...     y = np.exp(-2.0 * x)
...     return (1.0 - y) / (1.0 + y)
...
>>> grad_tanh = grad(tanh)       # Obtain its gradient function
>>> grad_tanh(1.0)               # Evaluate the gradient at x = 1.0
0.41997434161402603
>>> (tanh(1.0001) - tanh(0.9999)) / 0.0002  # Compare to finite differences
0.41997434264973155
```

Autodiff in Python: Autograd module

```
>>> from autograd import elementwise_grad as egrad  # for functions that vectorize over inputs
>>> import matplotlib.pyplot as plt
>>> x = np.linspace(-7, 7, 200)
>>> plt.plot(x, tanh(x),
...          x, egrad(tanh)(x),                                     # first  derivative
...          x, egrad(egrad(tanh))(x),                              # second derivative
...          x, egrad(egrad(egrad(tanh)))(x),                       # third  derivative
...          x, egrad(egrad(egrad(egrad(tanh))))(x),                # fourth derivative
...          x, egrad(egrad(egrad(egrad(egrad(tanh)))))(x),         # fifth  derivative
...          x, egrad(egrad(egrad(egrad(egrad(egrad(tanh))))))(x))  # sixth  derivative
>>> plt.show()
```



Autodiff in Python: PyTorch

PyTorch

```
import torch
x = torch.ones(2, 2, requires_grad=True)
print(x)

Out:
tensor([[1., 1.],
        [1., 1.]], requires_grad=True)

y = x + 2
z = y * y * 3
out = z.mean()
print(z, out)

Out:
tensor([[27., 27.],
        [27., 27.]], grad_fn=<MulBackward0>) tensor(27., grad_fn=<MeanBackward1>)

out.backward()

Out:
tensor([[4.5000, 4.5000],
        [4.5000, 4.5000]])
```

Navier-Stokes equations

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \mathbf{u} + \mathbf{f}$$

$$\frac{\partial \rho}{\partial t} = -(\mathbf{u} \cdot \nabla)\rho + \kappa \nabla^2 \rho + S$$

Applications of autodiff

- Variational Inference
- Deep learning
- Numerical optimization
- Hamiltonian Monte Carlo

Hamiltonian Monte Carlo

$$\pi(q, p) = e^{-H(q,p)}$$

$$\begin{aligned} H(q, p) &= -\log \pi(p \mid q) - \log \pi(q) \\ &\equiv \quad K(p, q) \quad + \; V(q) \,. \end{aligned}$$

Hamiltonian Monte Carlo

$$\pi(q,p) = e^{-H(q,p)}$$

$$\begin{aligned} H(q,p) &= -\log \pi(p \mid q) - \log \pi(q) \\ &\equiv \quad K(p,q) \quad + \; V(q)\,. \end{aligned}$$

$$\begin{aligned} \frac{\mathrm{d}q}{\mathrm{d}t} &= +\frac{\partial H}{\partial p} = \frac{\partial K}{\partial p} \\ \frac{\mathrm{d}p}{\mathrm{d}t} &= -\frac{\partial H}{\partial q} = -\frac{\partial K}{\partial q} - \frac{\partial V}{\partial q} \end{aligned}$$

Hamiltonian Monte Carlo

- Hamiltonian Monte Carlo (HMC) is vastly more efficient than Metropolis-Hastings or similar samplers
- It is the only method which works in very high-dimensional parameter spaces
- However, HMC requires gradients of log-probability w.r.t. to all of the parameters
- Gradients usually provided by autodiff
- Popular probabilistic modeling frameworks such as Stan and PyMC3 include autodiff libraries

Stan

PyMC3

Further reading

- "A review of automatic differentiation and its efficient implementation" - Carles C. Margossian
- Ian Murray's MLPR course notes: http://www.inf.ed.ac.uk/teaching/courses/mlpr/2018/notes/
- "Automatic Differentiation and Cosmology Simulation" - https://bids.berkeley.edu/news/automatic-differentiation-and-cosmology-simulation
- http://www.autodiff.org/
- https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html