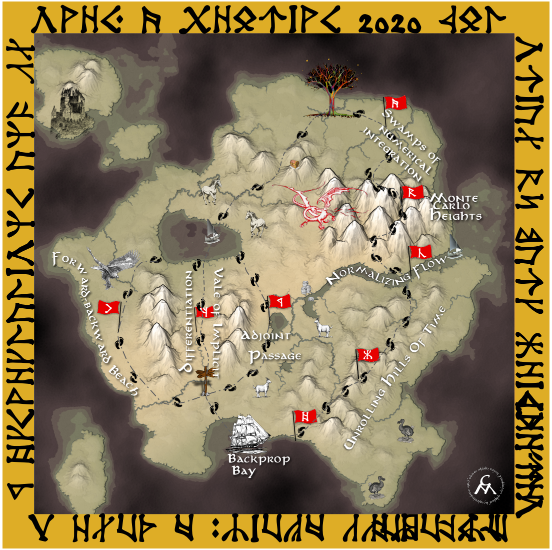
2020
Swamps of Numerical Integration
Monte Carlo Heights
Normalizing Flow
Unrolling Hills of Time
Backprop Bay
Forward-Backward Beach
Differentiation
Vale of Implicit


2020
Swamps of Numerical Integration
Monte Carlo Heights
Normalizing Flow
Adjoint Passage
Vale of Implicit
Differentiation
Forward-backward Beach
Unrolling Hills of Time
Backprop Bay

Method of Adjoints

Cheng Soon Ong
Marc Peter Deisenroth

December 2020

Motivation

Automatic differentiation

Augment each variable (for example a) with an **adjoint** variable $\overleftarrow{a}$ to form an adjoint pair $(a, \overleftarrow{a})$.

Adjoint, I've heard that somewhere else...

Linear dynamical system

Consider a dynamical system with state variable $\boldsymbol{x}_t$ and input variable $\boldsymbol{u}_t$. Assume that state evolves according to linear dynamics

$$\boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t \quad \text{for} \quad t = 0, 1, \ldots$$

where $(\boldsymbol{A}, \boldsymbol{B})$ are known state evolution matrices.

Find good control inputs

Find a sequence of inputs $\boldsymbol{u}_t$ that minimizes some quadratic cost over the trajectory, for a given $\boldsymbol{x}_0$:

$$\begin{aligned} \text{min}_{\boldsymbol{u}_t, \boldsymbol{x}_t} \quad & \frac{1}{2}\boldsymbol{x}_N^\top \boldsymbol{S} \boldsymbol{x}_{N+1} + \frac{1}{2}\sum_{t=0}^{N} \boldsymbol{x}_t^\top \boldsymbol{Q} \boldsymbol{x}_t + \boldsymbol{u}_t^\top \boldsymbol{R} \boldsymbol{u}_t, \\ \text{subject to} \quad & \boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t, \quad \text{for} \quad t = 0, 1, \dots, N. \end{aligned}$$

Lagrangian

The Lagrangian has the form (with Lagrange multiplier $\boldsymbol{\lambda}$)

$$\mathfrak{L}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{x}_N^\top \boldsymbol{S}\boldsymbol{x}_{N+1} + \frac{1}{2}\sum_{t=0}^{N} \boldsymbol{x}_t^\top \boldsymbol{Q}\boldsymbol{x}_t + \boldsymbol{u}_t^\top \boldsymbol{R}\boldsymbol{u}_t - \boldsymbol{\lambda}_t^\top (\boldsymbol{x}_{t+1} - \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t).$$

To satisfy $\nabla_{\boldsymbol{x}_t}\mathfrak{L} = 0$ we solve for the update equation for $\boldsymbol{\lambda}$,

$$\boldsymbol{\lambda}_{t-1} = \boldsymbol{A}^\top \boldsymbol{\lambda}_t + \boldsymbol{Q}\boldsymbol{x}_t.$$

This is called the **adjoint dynamics** (with initial condition $\boldsymbol{\lambda}_N = \boldsymbol{S}\boldsymbol{x}_{N+1}$).

Caching. In reverse.

We build $\boldsymbol{\lambda}_{t-1}$ by multiplying $\boldsymbol{\lambda}_t$ with $\boldsymbol{A}$

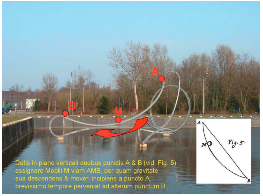
What is in a name?

In dynamical systems, the Lagrange multipliers are called **costates** or **adjoint variables** and the dual optimization problem is called the **adjoint problem**.

"Adjoint functors arise everywhere",
– Saunders Mac Lane, 1998, Categories for the Working Mathematician.

Pointers to literature

- Carathéodory's royal road (Pesch, 2012)
- Pontryagin's maximum principle (Pontryagin et al., 1964; Ohsawa, 2015)
- Book on optimal control (Anderson and Moore, 2007)
- Neural network view (Chen et al., 2018; Finlay et al., 2020)
- ICERM workshop on Scientific Machine Learning
  `https://icerm.brown.edu/events/ht19-1-sml/`



Sculpture at Groningen,
via Pesch & Plail

Find good control inputs

Find a sequence of inputs $\boldsymbol{u}_t$ that minimizes some quadratic cost over the trajectory, for a given $\boldsymbol{x}_0$:

$$\begin{aligned} \text{min}_{\boldsymbol{u}_t, \boldsymbol{x}_t} \quad & \frac{1}{2}\boldsymbol{x}_N^\top \boldsymbol{S}\boldsymbol{x}_{N+1} + \frac{1}{2}\sum_{t=0}^{N} \boldsymbol{x}_t^\top \boldsymbol{Q}\boldsymbol{x}_t + \boldsymbol{u}_t^\top \boldsymbol{R}\boldsymbol{u}_t, \\ \text{subject to} \quad & \boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t, \quad \text{for} \quad t = 0, 1, \dots, N. \end{aligned}$$

We consider the more general function

$$x^* = \begin{array}{c} \text{argmin}_x F(u, x) \\ \text{subject to } G(u, x) = 0 \end{array} = \begin{array}{c} \text{argmin}_x F(g(x), x) \\ \text{subject to } G(u, x) = 0\,. \end{array}$$

Recall: Jacobian

Definition (Jacobian)

The collection of all first-order partial derivatives of a vector-valued function
$\boldsymbol{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$
is called the **Jacobian**.

$$\boldsymbol{J} = \nabla_{\boldsymbol{x}} \boldsymbol{f} = \frac{d\boldsymbol{f}(\boldsymbol{x})}{d\boldsymbol{x}} = \begin{bmatrix} \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial x_n} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\boldsymbol{x})}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\boldsymbol{x})}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$J(i, j) = \frac{\partial f_i}{\partial x_j} .$$

Recall: Automatic differentiation

Backward pass of reverse mode automatic differentiation

Given a function $G(w) := g(f(e(w)))$, with intermediate variables x, y and final output z.

$$w \longrightarrow e(\cdot) \longrightarrow x \longrightarrow f(\cdot) \longrightarrow y \longrightarrow g(\cdot) \longrightarrow z$$

Input adjoint $=$ output adjoint $\times$ forward partials

$$\underbrace{\frac{dz}{dw}}_{\overleftarrow{w}} = \underbrace{\left(\frac{dz}{dy}\frac{dy}{dx}\right)}_{\overleftarrow{x}} \frac{dx}{dw} \qquad \text{(reverse mode)}$$

Vectorized computation

Efficient compute

Exploit computational architecture using the **Vector Jacobian Product**.

Assume that function $e : \mathbb{R}^n \to \mathbb{R}^m$

Input adjoint $=$	output adjoint$\times$	forward partials
$\overleftarrow{w} =$	$\overleftarrow{x} \times$	$\dfrac{\partial x}{\partial w}$
$\mathbb{R}^{1\times n}$	$\mathbb{R}^{1\times m}$	$\mathbb{R}^{m\times n}$

Constrained optimization

Consider the general optimization problem with two (potentially vector) variables:

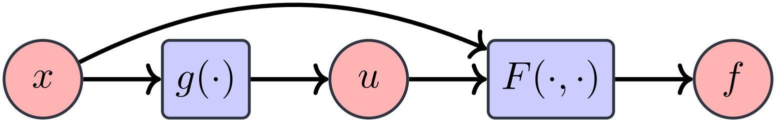
$$
\begin{aligned}
x^* = \quad & \mathrm{argmin}_x F(g(x), x) \\
& \text{subject to } G(u, x) = 0\,.
\end{aligned}
$$

Constrained optimization

Consider the general optimization problem with two (potentially vector) variables:

$$x^* = \begin{aligned} &\text{argmin}_x F(g(x), x) \\ &\text{subject to } G(u, x) = 0\,. \end{aligned}$$

Think about how the information flows



Recall: Implicit function theorem

The implicit function theorem **provides conditions** under which we can write $G(x, y) = 0$ as $y = g(x)$, and also conditions when $g'(x) = -\frac{G_x}{G_y}$.

Hand wavy intuition

Given three variables (could be any topology) x, y, z and

$$G(x, y) = z\,,$$

What are the conditions for the following to be well behaved?

$$\begin{aligned} g(x) &= y \\ G(x, g(x)) &= z\,. \end{aligned}$$

Solve for gradient of constraint

Differentiate G with respect to x, using the chain rule,

$$\frac{dG}{dx} = \frac{\partial G}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial G}{\partial x}\frac{\partial x}{\partial x}\,.$$

Solve for stationarity conditions, setting $\frac{dG}{dx} = 0$,

$$\frac{\partial u}{\partial x} = -\left(\frac{\partial G}{\partial u}\right)^{-1}\frac{\partial G}{\partial x}\,.$$

Chain rule on objective function F

Differentiate F with respect to x, using the chain rule,

$$\frac{dF}{dx} = \frac{\partial F}{\partial x}\frac{\partial x}{\partial x} + \frac{\partial F}{\partial u}\frac{\partial u}{\partial x}\,.$$

Substituting the solution of $\frac{\partial u}{\partial x}$ into $\frac{dF}{dx}$, we obtain

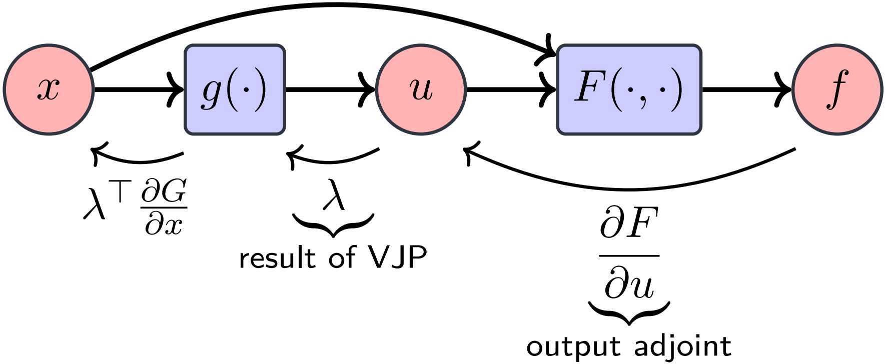
$$\frac{dF}{dx} = \frac{\partial F}{\partial x} - \underbrace{\frac{\partial F}{\partial u}\left(\frac{\partial G}{\partial u}\right)^{-1}}_{=-\lambda^\top}\frac{\partial G}{\partial x}\,.$$

Observe that λ can be computed by a vector Jacobian product.

Identify Lagrange multiplier λ

Rewrite in terms of λ.

$$\frac{dF}{dx} = \frac{\partial F}{\partial x} + \lambda^\top \frac{\partial G}{\partial x} \quad \text{where} \quad \lambda = \frac{\partial F}{\partial u}\left(\frac{\partial G}{\partial u}\right)^{-1}$$



Adjoint = Lagrange multiplier in method of adjoints

We consider the general function

$$x^* = \begin{array}{c}\text{argmin}_x F(u,x) \\ \text{subject to } G(u,x)=0\end{array} = \begin{array}{c}\text{argmin}_x F(g(x),x) \\ \text{subject to } G(u,x)=0\,.\end{array}$$

From the implicit function theorem

$$\frac{dF}{dx} = \frac{\partial F}{\partial x} + \lambda^\top \frac{\partial G}{\partial x} \quad \text{where} \quad \lambda = \frac{\partial F}{\partial u}\left(\frac{\partial G}{\partial u}\right)^{-1}$$

Alternatively, by the method of Lagrange

$$\nabla_x \mathfrak{L} = \nabla_x F(x,u) + \boldsymbol{\lambda}^\top \nabla_x G(x,u)$$

▶▶▶ **adjoint dynamics** $(\boldsymbol{\lambda}_{t-1} = \boldsymbol{A}^\top \boldsymbol{\lambda}_t + \boldsymbol{Q}\boldsymbol{x}_t)$.

Summary

- Method of adjoints studied in optimal control
- ... adjoint state method, Pontryagin's principle
- Vector Jacobian products for efficient computation
- Adjoint variable in autodiff = Lagrange multiplier

Backpropagation is just ...

the method of adjoints

References

Anderson, B. D. O. and Moore, J. B. (2007). *Optimal Control: Linear Quadratic Methods*. Dover.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*.

Finlay, C., Jacobsen, J.-H., Nurbekyan, L., and Oberman, A. M. (2020). How to train your neural ODE: the world of Jacobian and kinetic regularization.

Lane, S. M. (1998). *Categories for the Working Mathematician*. Springer, 2nd edition.

Ohsawa, T. (2015). Contact Geometry of the Pontryagin Maximum Principle. *Automatica*, 55(2015):1–5.

Pesch, H. J. (2012). In *Optimization Stories*.

Pontryagin, L. S., Boltyanskii, V. G., Gamkrelidze, R. V., and Mishchenko, E. F. (1964). *The Mathematical Theory of Optimal Processes*. Pergamon Press.