Optimization Oracles for Chains of Computations

Vincent Roulet, Zaid Harchaoui
Department of Statistics, University of Washington, Seattle

April 19, 2020

Chains of Computations

Definition (Chain of computations)

A chain ψ of τ computations parametrized by $w_{1:\tau} = (w_1, \ldots, w_\tau)$ is defined by τ elementary functions ϕ_t such that for $x_0 \in \mathbb{R}^{d_0}$

$$\psi(x_0, w_{1:\tau}) = x_\tau$$
$$\text{where} \quad x_t = \phi_t(x_{t-1}, w_t) \quad \text{for } t \in \{1, \ldots, \tau\}$$

Chains of Computations

Definition (Chain of computations)

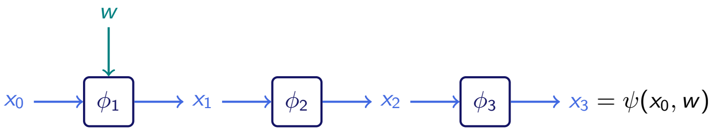
A chain ψ of τ computations parametrized by $w_{1:\tau} = (w_1, \ldots, w_\tau)$ is defined by τ elementary functions ϕ_t such that for $x_0 \in \mathbb{R}^{d_0}$

$$\psi(x_0, w_{1:\tau}) = x_\tau$$
$$\text{where} \quad x_t = \phi_t(x_{t-1}, w_t) \quad \text{for } t \in \{1, \ldots, \tau\}$$

Example

Logistic function $\psi(x_0, w) = \log(1 + \exp(-x_0^\top w))$

1. $\phi_1(x_0, w) = x_0^\top w$,
2. $\phi_2(x_1) = 1 + \exp(x_1)$,
3. $\phi_3(x_2) = \log(x_2)$.



Chains of Computations

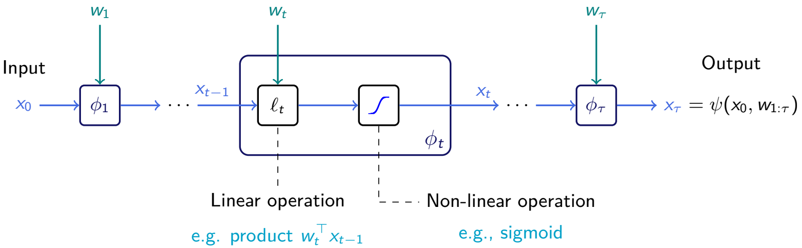
Definition (Chain of computations)

A chain ψ of τ computations parametrized by $w_{1:\tau} = (w_1, \ldots, w_\tau)$ is defined by τ elementary functions ϕ_t such that for $x_0 \in \mathbb{R}^{d_0}$

$$\psi(x_0, w_{1:\tau}) = x_\tau$$
$$\text{where} \quad x_t = \phi_t(x_{t-1}, w_t) \quad \text{for } t \in \{1, \ldots, \tau\}$$

Example

Deep networks



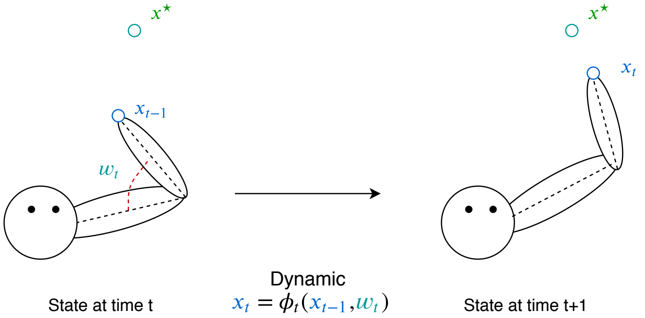
Chains of Computations

Example

Control problem

$$\min_{w_1,\ldots,w_\tau} \quad \|x_\tau - x^\star\|_2^2 + \sum_{t=1}^{\tau} \lambda \|u_t\|_2^2$$

$$\text{sujet à} \quad x_{t+1} = \phi_t(x_t, w_{t+1}), \qquad x_0 = \hat{x}_0$$



Optimization Problem

Objective

Given a chain of computations ψ, convex functions h h_i, g_t and $x \in \mathbb{R}^{d_0}$

$$\min_{w_{1:\tau}} \; h(\psi(x, w_{1:\tau})) + \sum_{t=1}^{\tau} g_t(w_t)$$

Optimization Problem

Objective

Given a chain of computations ψ, convex functions h h_i, g_t and $x_i \in \mathbb{R}^{d_0}$

$$\min_{w_{1:\tau}} \; \frac{1}{n}\sum_{i=1}^{n} h_i(\psi(x_i, w_{1:\tau})) + \sum_{t=1}^{\tau} g_t(w_t)$$

Optimization Problem

Objective

Given a chain of computations ψ, convex functions h h_i, g_t and $x_i \in \mathbb{R}^{d_0}$

$$\min_{w_{1:\tau}} \frac{1}{n}\sum_{i=1}^{n} h_i(\psi(x_i, w_{1:\tau})) + \sum_{t=1}^{\tau} g_t(w_t)$$

Questions

How can we decompose (i) gradients, (ii) Gauss-Newton, (iii) Newton, ((iv) risk-sensitive gradients), (v) proximal points oracles for

$$f(w_{1:\tau}) = h(\psi(x, w_{1:\tau})) + \sum_{t=1}^{\tau} g_t(w_t)$$

into the dynamical structure of $\psi(x, \cdot)$?

Oracles Decomposition

Approach

Oracles are defined by subproblems

1. Decompose theoretically the sub-problems into the chain of computations
2. Get an efficient implementation of the subproblems

Oracles Decomposition

Approach
Oracles are defined by subproblems

1. Decompose theoretically the sub-problems into the chain of computations
2. Get an efficient implementation of the subproblems

Model minimization oracles
Given a model of f at x s.t. $m_f(y;x) \approx f(y)$ (e.g. $|m_f(y;x) - f(y)| \leq L\|x-y\|_2^2$)
Oracle defined as (with γ stepsize)

$$\mathcal{O}f(x) = \arg\min_y m_f(x+y;x) + \frac{1}{2\gamma}\|y\|_2^2$$

Examples:

1. *Gradient* $m_f = \ell_f$ (linear approx.)
2. For $f = h \circ \psi + g$, *Gauss-Newton* $m_f = q_h \circ \ell_\psi + q_g$ (mixed approx.)
3. *Newton* $m_f = q_f$ (quadratic approx.)

Gradient Oracle

Gradient oracle decomposition
For

$$f(w_{1:\tau}) = h(\psi(x, w_{1:\tau})) + \sum_{t=1}^{\tau} g_t(w_t) \qquad \text{with} \quad \begin{aligned} \psi(x, w_{1:\tau}) &= x_\tau \\ x_{t+1} &= \phi_t(x_t, w_t) \\ x_0 &= x \end{aligned}$$

Gradient Oracle

Gradient oracle decomposition

For

$$f(w_{1:\tau}) = h(\psi(x, w_{1:\tau})) + \sum_{t=1}^{\tau} g_t(w_t) \qquad \text{with} \quad \begin{aligned} \psi(x, w_{1:\tau}) &= x_\tau \\ x_{t+1} &= \phi_t(x_t, w_t) \\ x_0 &= x \end{aligned}$$

gradient is given by $-\gamma \nabla f(w_{1:\tau}) = {v_{1:\tau}}^*$ solution of

$$\begin{aligned} \min_{\substack{v_1,\dots,v_\tau \\ y_0,\dots,y_\tau}} \quad & \overbrace{\nabla h(x_\tau)^\top y_\tau}^{\text{linearization of the objective}} + \overbrace{\sum_{t=1}^{\tau} \nabla g_t(w_t)^\top v_t}^{\text{linearization of the penalty}} + \frac{1}{2\gamma} \sum_{t=1}^{\tau} \|v_t\|_2^2 \\ \text{subject to} \quad & y_t = \underbrace{\nabla_x \phi_t(w_t, x_{t-1})^\top y_{t-1} + \nabla_w \phi_t(w_t, x_{t-1})^\top v_t}_{\text{linearization of the computations}}, \quad y_0 = 0 \end{aligned}$$

Gradient Oracle

Gradient oracle decomposition
For

$$f(w_{1:\tau}) = h(\psi(x, w_{1:\tau})) + \sum_{t=1}^{\tau} g_t(w_t) \qquad \text{with} \quad \begin{aligned} \psi(x, w_{1:\tau}) &= x_\tau \\ x_{t+1} &= \phi_t(x_t, w_t) \\ x_0 &= x \end{aligned}$$

gradient is given by $-\gamma \nabla f(w_{1:\tau}) = {v_{1:\tau}}^*$ solution of

$$\min_{\substack{v_1,\ldots,v_\tau \\ y_0,\ldots,y_\tau}} \quad {\tilde{h}_\tau}^\top y_\tau + \sum_{t=1}^{\tau} {\tilde{g}_t}^\top v_t + \frac{1}{2\gamma}\|v_t\|_2^2$$

$$\text{subject to} \quad y_t = {\Phi_t^x}^\top y_{t-1} + {\Phi_t^w}^\top v_t, \quad y_0 = 0$$

Quadratic problem with linear dynamical constraints

Dual Viewpoint

With dual variables λ_t,

$$\min_{\substack{v_1,\dots,v_\tau \\ y_1,\dots,y_\tau}} \sup_{\lambda_1,\dots,\lambda_\tau} \tilde{h}_\tau^\top y_\tau + \sum_{t=1}^{\tau} \tilde{g}_t^\top v_t + \frac{1}{2\gamma}\|v_t\|_2^2 + \sum_{t=1}^{\tau} \lambda_t^\top (\Phi_t^{x\top} y_{t-1} + \Phi_t^{w\top} v_t - y_t) + \lambda_0^\top y_0$$

Swapping $\min_{y_0,\dots,y_\tau}$ and $\sup_{\lambda_1,\dots,\lambda_\tau}$, after minimization in y_t, we get

$$\min_{w_1,\dots,w_\tau} \sup_{\lambda_0,\dots,\lambda_\tau} \sum_{t=1}^{\tau} (\tilde{g}_t + \Phi_t^w \lambda_t)^\top v_t + \frac{1}{2\gamma}\|v_t\|_2^2$$

$$\text{subject to} \quad \lambda_\tau = \tilde{h}_\tau, \quad \lambda_{t-1} = \Phi_t^x \lambda_t \tag{1}$$

Gradient given by $-\gamma \nabla f(w_{1:\tau}) = {v_{1:\tau}}^*$ with

$$v_t^* = -\gamma(\tilde{g}_t + \Phi_t^w \lambda_t) \tag{2}$$

Algorithm (Automatic-Differentiation)

1. Compute dual variables by λ_t by (1)
2. Output gradient by (2)

Dynamic Programming Viewpoint

Define optimal cost starting from $\hat{y}_t$ at time t,

$$\text{cost}(\hat{y}_{t-1}) = \min_{\substack{v_t,\ldots,v_\tau \\ y_{t-1},\ldots y_\tau}} \quad \tilde{h}_\tau^\top y_\tau + \sum_{s=t}^{\tau} \tilde{g}_s{}^\top v_s + \frac{1}{2\gamma}\|v_s\|_2^2$$

$$\text{subject to} \quad y_s = A_s{}^\top y_{s-1} + B_s{}^\top v_s, \quad y_{t-1} = \hat{y}_{t-1} \qquad \text{for } s = t, \ldots, \tau$$

Dynamic Programming Viewpoint

Define optimal cost starting from $\hat{y}_t$ at time t,

$$\text{cost}(\hat{y}_{t-1}) \quad = \quad \min_{v_t} \quad \tilde{g}_t^\top v_t + \frac{1}{2\gamma}\|v_t\|_2^2 + \text{cost}({\Phi_s^x}^\top y_{t-1} + {\Phi_t^w}^\top v_t)$$

Dynamic Programming Viewpoint

Define optimal cost starting from $\hat{y}_t$ at time t,

$$\text{cost}(\hat{y}_{t-1}) \quad = \quad \min_{v_t} \quad \tilde{g}_t^\top v_t + \frac{1}{2\gamma}\|v_t\|_2^2 + \text{cost}({\Phi_s^x}^\top y_{t-1} + {\Phi_t^w}^\top v_t)$$

Can show recursively

$$\text{cost}(y_t) = {\lambda_t}^\top y_{t-1}$$

$$\text{where} \quad \lambda_\tau = \tilde{h}_\tau, \quad \lambda_{t-1} = \Phi_t^x \lambda_t$$

Optimal controls

$$\begin{aligned} v_t^* &= \arg\min_{v_t} \tilde{g}_t^\top v_t + \frac{1}{2\gamma}\|v_t\|_2^2 + \text{cost}({\Phi_s^x}^\top y_{t-1} + {\Phi_t^w}^\top v_t) \\ &= -\gamma(\tilde{g}_t + \Phi_t^w \lambda_t) \end{aligned}$$

Automatic-Differentiation

Viewpoints

1. *Traditional*: chain rule, Lagrangian trick, . . .
2. *Dual*: (i) Highlight "co-states" λ_t as dual variables of the linearized pb,
   (ii) Useful to generalize to e.g. proximal points
3. *Dynamic Programming:* Can tackle gradients, Gauss-Newton, Newton

Consequences

1. Baur-Strassen's theorem (?),

   "computing a derivative is up to a constant factor
   more expansive than computing the function"

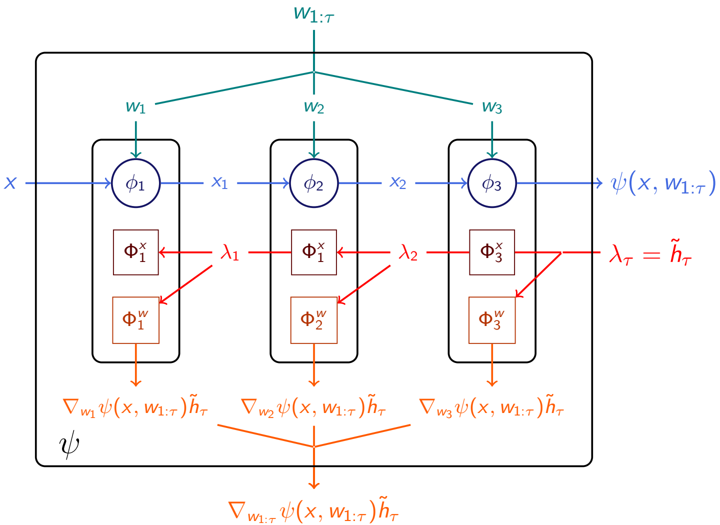
2. Access to, by machine learning libraries (e.g. Pytorch (?)),

$$\lambda \to \nabla\psi(w_{1:\tau}, x)\lambda$$

We do not store $\nabla\psi(w_{1:\tau}, x)$ but have access to the linear form
by only storing $(\nabla\phi_t)_{t=1}^{\tau}$

Automatic differentiation

$$\nabla f(w_{1:\tau}) = \nabla_{w_{1:\tau}} \psi(x, w_{1:\tau}) \nabla h(x_\tau)$$



Gradient, Gauss-Newton, Newton by Dynamic Programming

On a point $w_{1:\tau} \in \mathbb{R}^p$, given a step-size γ, for an objective of the form $h \circ \psi + g$,

Gradient
A *gradient* step is defined as

$$w_{1:\tau}^{+} = \operatorname*{arg\,min}_{v_{1:\tau}} \ell_{h\circ\psi}(v_{1:\tau}; w_{1:\tau}) + \ell_g(v_{1:\tau}; w_{1:\tau}) + \frac{1}{2\gamma}\|v_{1:\tau} - w_{1:\tau}\|_2^2,$$

Gauss-Newton
A (regularized generalized) *Gauss-Newton* step is defined as

$$w_{1:\tau}^{+} = \operatorname*{arg\,min}_{v_{1:\tau}} q_h(\ell_\psi(v_{1:\tau}; w_{1:\tau}); \psi(w_{1:\tau})) + q_g(v_{1:\tau}; w_{1:\tau}) + \frac{1}{2\gamma}\|v_{1:\tau} - w_{1:\tau}\|_2^2,$$

Newton
A (regularized) *Newton* step is defined as

$$w_{1:\tau}^{+} = \operatorname*{arg\,min}_{v_{1:\tau}} q_{h\circ\psi}(v_{1:\tau}; w_{1:\tau}) + q_g(v_{1:\tau}; w_{1:\tau}) + \frac{1}{2\gamma}\|v_{1:\tau} - w_{1:\tau}\|_2^2.$$

Gradient, Gauss-Newton, Newton by Dynamic Programming

Proposition ((???))

Gradient, Gauss-Newton, Newton steps amount to solve

$$\min_{\substack{v_1,\ldots,v_\tau \\ y_0,\ldots,y_\tau}} \quad \sum_{t=1}^{\tau} \frac{1}{2} y_t^\top P_t y_t + p_t^\top y_t + y_{t-1}^\top R_t v_t + \frac{1}{2} v_t^\top Q_t v_t + q_t^\top v_t + \frac{1}{2\gamma}\|v_t\|_2^2$$

$$\text{subject to} \quad y_t = \Phi_t^{x\top} t y_{t-1} + \Phi_t^{w\top} v_t \qquad \text{for} \quad t \in \{1,\ldots,\tau\},$$

$$y_0 = 0,$$

Example

For Newton steps, defining

$$\lambda_\tau = \nabla h(\psi(w^{(k)})), \quad \lambda_{t-1} = \nabla_{x_{t-1}} \phi_t(w_t, x_{t-1}) \lambda_t \quad \text{for } t \in \{1,\ldots,\tau\},$$

we have

$$P_\tau = \nabla^2 h(\psi(w^{(k)})), P_{t-1} = \nabla^2_{x_{t-1}x_{t-1}} \phi_t(w_t, x_{t-1})[\cdot,\cdot,\lambda_t] \quad \text{for } t \in \{1,\ldots,\tau\},$$

$$R_t = \nabla^2_{x_{t-1}w_t} \phi_t(w_t, x_{t-1})[\cdot,\cdot,\lambda_t], Q_t = \nabla^2 g_t(w_t) + \nabla^2_{w_t w_t} \phi_t(w_t, x_{t-1})[\cdot,\cdot,\lambda_t].$$

Implementation of Gauss-Newton by Automatic Differentiation

Dual of Gauss-Newton step

1. Formulation

$$
\begin{aligned}
&\min_{\lambda} \quad \tilde{q}_h^\star(\lambda) + \tilde{q}_g^\star(-\nabla\psi(w_{1:\tau}, x)\lambda), \\
\text{where} \quad &\tilde{q}_h(y) = q_h(\psi(w_{1:\tau}) + y; \psi(w_{1:\tau})), \\
&\tilde{q}_g(z) = q_g(w_{1:\tau} + z; w_{1:\tau}) + \|z\|_2^2/2
\end{aligned}
$$

2. Gauss-Newton-step reads $z^{(k+1)} = w_{1:\tau} + \nabla\tilde{q}_g^\star(-\nabla\psi(w_{1:\tau})\lambda^*)$
3. Can be solved by $2q+1$ calls to an automatic differentiation procedure where q is the output dimension of ψ.

Consequences

1. All those steps are linear quadratic control problems
2. Can be solved by dynamic programming with a **linear** complexity w.r.t. τ

Implementation

1. Compute in a backward pass, cost-to-go functions as quadratics,
2. Store solutions at each step as $v_t^*(y_{t-1}) = K_t y_{t-1} + k_t$
3. Solve subproblems in a forward pass by,

$$\begin{aligned} y_0 &= 0 \\ v_t^* &= K_t y_{t-1} + k_t \\ y_t &= \Phi_t^{x\top} y_{t-1} + \Phi_t^{w\top} w_t \quad \text{for } t = 1, \ldots \tau \end{aligned}$$

Actual Algorithms in Non-Linear Control

Differential Dynamic Programming (?)

1. *Idea:* Back-propagate quadratic approximations of Bellman's equation

$$\text{cost}(x_{t-1}) = \min_{y_t} g_t(w_t) + \text{cost}(\phi_t(x_t, w_t))$$

2. Resulting cost-to-go functions are similar to the ones for Newton's method but the forward pass reads

$$\begin{aligned} y_0 &= \hat{x}_0 \\ v_t^* &= K_t y_{t-1} + k_t \\ y_t &= \phi_t(y_{t-1}, w_t) \quad \text{for } t = 1, \ldots \tau \end{aligned}$$

Analysis ?

1. Can be analyzed as perturbed Newton (?)
2. Yet, better behavior in practice (?)
3. Can be seen as a recursive projected method on states (?)

Smoothness Considerations

Proposition (Automatic smoothness computations)

Assume computations ϕ_t to be ℓ_{ϕ_t} Lipschitz continuous and L_{ϕ_t} smooth,

1. *Upper bound on Lipschitz-continuity of ψ is given by $\ell_\psi = \ell_\tau$, where*

$$\ell_t = \ell_{\phi_t} + \ell_{t-1}\ell_{\phi_t}, \qquad \ell_0 = 0.$$

2. *Upper-bound of Smoothness of ψ is given by $L_\psi = L_\tau$, where*

$$L_t = L_{t-1}\ell_{\phi_t} + L_{\phi_t}(1 + \ell_{t-1})^2, \qquad L_0 = 0.$$

Automatic smoothness computations

Generalizes to smoothness estim. of deep networks on balls (?)

Get automatic smoothness comparisons of deep networks

Can be used to derive optimization convergence rates