Designing Automatic Differentiation Algorithms for Hessian Matrices

Robert Mansel Gower gowerrobert@gmail.com   Margarida P. Mello margarid@ime.unicamp.br

Introduction

Automatic Differentiation *AD* has had a lot of success in calculating exact gradient with the same time complexity as evaluating the underlying function. The need to efficiently calculate Hessian matrices is driven by the rising popularity of constraint optimization methods that employ second-order information.
We analyse the inherent symmetries involved in calculting the Hessian and a framework for demonstrating and designing Hessian AD algorithms. Using this framework we design a new competitive Hessian algorithm that takes full advantage of these symmetries.

Function and Gradient

Elemental Function $= \phi_i$
Coded evaluation rules + derivatives
$j \prec i \Longrightarrow$ need result of ϕ_j to evaluate ϕ_i
State Transformations $= \Phi_i$

$$\begin{aligned}\Phi_i : \quad & \mathbb{R}^{n+\ell} \to \mathbb{R}^{n+\ell} \\ & y \mapsto y - e_i y_i + \phi_i(y_j)_{j \prec i}\end{aligned}$$

Evaluation Procedure

Input: $(x_{1-n}, \ldots, x_0)$
for $i = 1, \ldots, \ell$
  $v_i = \phi_i(v_j)_{j \prec i}$
for $i = \ell, \ldots, 1$
  for each $j \prec i$
   $\bar{v}_j += \bar{v}_i \frac{\partial \phi_i}{\partial v_j}$.
Output: $y = v_\ell$, $\nabla f = (\bar{v}_i)_{1-n \ldots 0}$

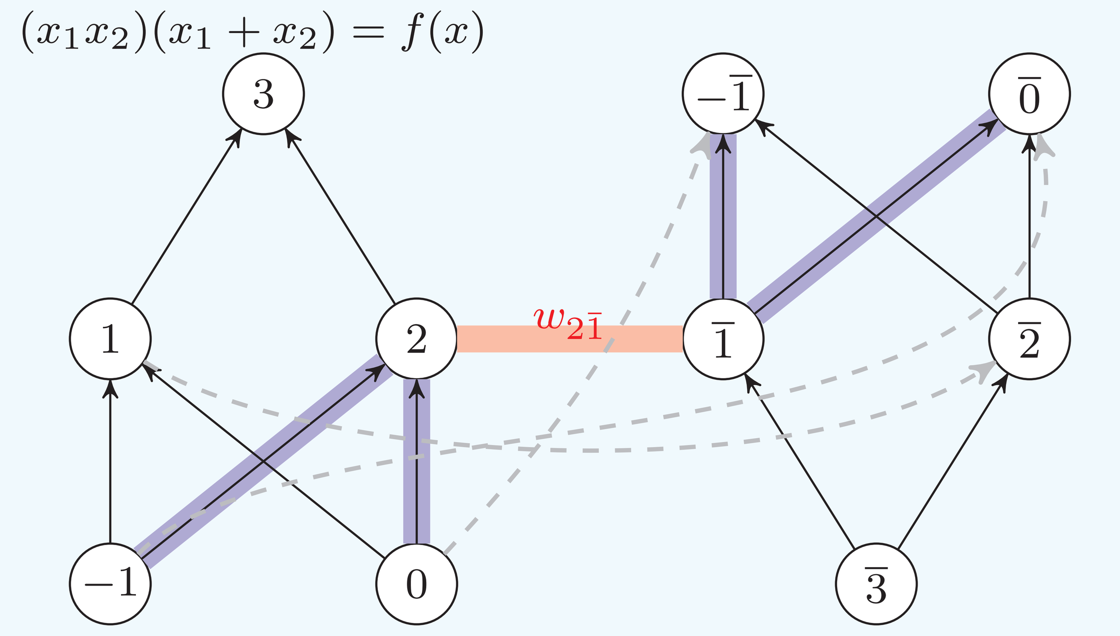
Block Evaluation

Input: $(x_{1-n}, \ldots, x_0)$
Intialization: $\mathrm{v} = P^T x$, $\bar{v} = e_\ell$
for $i = 1, \ldots, \ell$
  $\mathrm{v} \leftarrow \Phi_i(\mathrm{v})$
for $i = \ell, \ldots, 1$
  $\bar{v}^T \leftarrow \bar{v}^T \Phi_i'$.
Output: $y = \mathrm{v}_\ell$, $P\bar{v}$

$$\begin{aligned} f(x) &= e_\ell^T \Phi_\ell \circ \Phi_{\ell-1} \circ \cdots \circ \Phi_1(P^T x) \\ \nabla f(x)^T &= e_\ell^T \Phi_\ell' \Phi_{\ell-1}' \cdots \Phi_1'(P^T x) P^T \end{aligned}$$

$e_\ell^T = (0, \ldots, 0, 1) \qquad P = [I, 0, \ldots, 0]$.

Inherent Symmetry of The Gradient Graph



$$\begin{aligned} v_{-1} &= x_1 \\ v_0 &= x_2 \\ v_1 &= v_0 v_{-1} \\ v_2 &= (v_{-1} + v_0) \\ v_3 &= v_1 v_2 \end{aligned}$$

$$\begin{aligned} \bar{v}_3 &= 1 \\ \bar{v}_2 &= \bar{v}_3 v_1 \\ \bar{v}_1 &= \bar{v}_3 v_2 \\ \bar{v}_0 &= \bar{v}_2 1 + \bar{v}_1 v_{-1} \\ \bar{v}_{-1} &= \bar{v}_2 1 + \bar{v}_1 v_0 \end{aligned}$$

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \sum_{\{m,\bar{k}\} \in W} \; \sum_{p|\text{ from } i-n \text{ to } m} \left(\prod_{(i,j)\in p} w_{ij} \right) w_{m\bar{k}} \sum_{p|\text{ from } \bar{k} \text{ to } \overline{j-n}} \left(\prod_{(i,j)\in p} w_{ij} \right).$$

- Calculating $\partial^2 f / \partial x_i \partial x_j =$ accumulating all weights of paths from $(i-n)$ to $(\overline{j-n})$.
- Paths must take a crossing edge thus many paths intersect.
- Symmetric crossing edges $i \dashrightarrow \bar{j} \Longleftrightarrow \bar{i} \dashleftarrow j$
- $\{p \mid \text{ from } i-n \text{ to } m\}$ is one-to-one with $\{p \mid \text{ from } \bar{m} \text{ to } \overline{i-n}\}$

The Hessian Formula

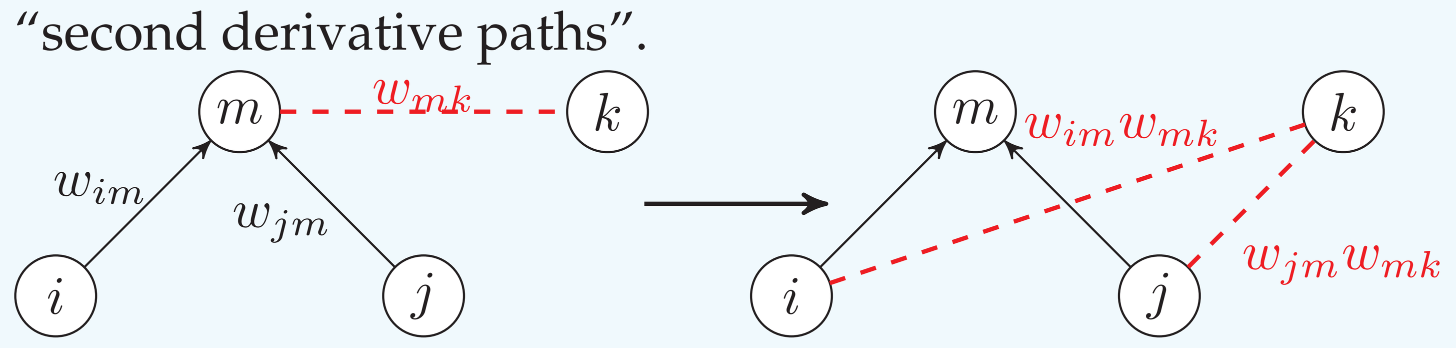
$$f'' = \boxed{\sum_{i=1}^{\ell} P(\Phi_1')^T(\Phi_2')^T \cdots (\Phi_{i-1}')^T((\bar{v}^i)^T \Phi_i'')\Phi_{i-1}' \cdots \Phi_2' \Phi_1' P^T}$$

$$(\bar{v}^i)^T = e_\ell^T \Phi_\ell' \cdots \Phi_{i+1}'$$

The problem of designing efficient Hessian algorithms is equivalent to designing an algorithm to efficiently calculate this formula. This strategy gives a broader perspective than deducing algorithms through differentiating gradient algorithms.

A New AD Hessian Algorithm

Observing the symmetry we have designed the algorithm `edge_pushing` that builds successive shortcuts for the “second derivative paths”.



Correctness can be proved succinctly using a block format of the algorithm and the Hessian formula. We have implemented `edge_pushing` embedded in ADOL-C as a driver.

Block format `edge_pushing`

Input: $x \in \mathbb{R}^n$
Intialization:
  $\bar{v} = e_\ell$
  $W \leftarrow 0 \in \mathbb{R}^{(n+\ell)^2}$
for $i = \ell, \ldots, 1-n$
  $W \leftarrow (\Phi_i')^T W \Phi_i'$
  $W \leftarrow W + \bar{v}^T \Phi_i''$
  $\bar{v}^T \leftarrow \bar{v}^T \Phi_i'$
Output: $f''(x) = PWP^T$

Test Results

Acyclic Coloring

names	# of runs
cosine	–
chainwoo	–
bc4	–
cragglevy	–
pspdoc	–
scon1dls	–
morebv	–
augmlagn	–
lminsurf	–
brybnd	–
arwhead	2
nondquar	3
sinquad	37
bdqrtic	3
noncvxu2	–
ncvxbqp1	–

Averages

	acyclic		e_p
	1st	2nd	
(s)	697.6	18.12	98.41

	star		e_p
	1st	2nd	
(s)	637.8	1.196	98.41

`e_p` $\equiv$ `edge_pushing`

dimension of problems $= 10^5$.

Star Coloring

names	# of runs
cosine	1239
chainwoo	6893
bc4	1257
cragglevy	772
pspdoc	–
scon1dls	–
morebv	–
augmlagn	–
lminsurf	–
brybnd	–
arwhead	3
nondquar	6
sinquad	15
bdqrtic	8
noncvxu2	–
ncvxbqp1	–

We compare `edge_pushing` to Gebremedhin *et al* 2005-2009 methods that combine graph coloring and AD Hessian-vector products on 16 hand picked CUTE examples.

Conclusions, Future Work

Graph and algebraic points of view converge in a new framework for proving the correctness of existing Hessian algorithms and designing new ones. It lead to the development of `edge_pushing`, a new algorithm that is truly reverse in nature and efficiently exploits symmetry. Computational tests with problems from CUTE show the potential of the new algorithm, which was faster than state-of-the-art algorithms in 12 out of 16 hand picked problems with distinct Hessian matrices.

References

[1] A. H. Gebremedhin and A. Tarafdar and A. Pothen and A. Walther. "Efficient Computation of Sparse Hessians Using Coloring and Automatic Differentiation". INFORMS J. on Computing, 2009.

[2] A. Griewank and A. Walther "Evaluating derivatives: principles and techniques of algorithmic differentiation". publisher SIAM, 2000