AUTOMATIC DIFFERENTIATION

P. CASPERS

First Version July 12, 2015 - This Version August 8, 2015
INCOMPLETE DRAFT

Abstract. We summarize the ideas underlying automatic differentiation.

Contents

1. Computational Graph

We consider the computation of a function $f : \mathbb{R}^n \to \mathbb{R}^m$ with independent variables $x_1, \ldots, x_n$ and dependent variables $y_1, \ldots, y_m$. The ultimate goal is to compute the Jacobian

$$J = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & & \cdots & & \frac{\partial y_1}{\partial x_n} \\ & \ddots & & & \\ \vdots & & \frac{\partial y_i}{\partial x_j} & & \vdots \\ & & & \ddots & \\ \frac{\partial y_m}{\partial x_1} & & \cdots & & \frac{\partial y_m}{\partial x_n} \end{pmatrix} \tag{1.1}$$

We view the function f as a composite of elementary operations

$$u_k = \Phi_k(\{u_\kappa\}_{(\kappa,k)\in\Lambda}) \tag{1.2}$$

for $k > n$ where we set $u_k = x_k$ for $k = 1, \ldots, n$ (i.e. we reserve these indices for the start values of the computation) and $u_k = y_k$ for $k = K - m + 1, \ldots, K$ (i.e. these are the final results of the computation). The notation should suggest that u_k depends on prior results u_κ with (κ, k) in some index set Λ. Note that if $(k, l) \in \Lambda$ this refers to a direct dependency of u_l on u_k, i.e. if u_k depends on u_j, but u_j does not enter the calculation of u_k directly then $(j, l) \notin \Lambda$.

Date: August 8, 2015.

We can view the computation chain as a directed graph with vertices u_k and edges (k,l) if $(k,l) \in \Lambda$. There are no circles allowed in this graph and it consists of K vertices.

We write $|i,j|$ for the length of the longest path from u_i to u_j and call that number the distance from i to j. If u_j is not reachable from u_i we set $|i,j| = \infty$.

2. Forward mode

we can compute a partial derivative $\partial u_m/\partial u_k$ using the chain rule

$$\frac{\partial u_m}{\partial u_k} = \sum_{l|(l,m)\in\Lambda} \frac{\partial u_m}{\partial u_l}\frac{\partial u_l}{\partial u_k} \tag{2.1}$$

This suggest a forward propagation scheme: We start at the initial nodes $u_1, ..., u_n$. For all nodes u_l with maximum distance 1 from all of these nodes we compute

$$c_l = \sum_{i=1,\dots,n} \frac{\partial u_l}{\partial u_i} c_i \tag{2.2}$$

where we can choose c_i for $i = 1, \dots, n$ freely at this stage. This assigns the dot product of the gradient of u_l w.r.t. $x_1, \dots, x_n$ and $(c_1, \dots, c_n)$ to the node u_l. If we choose $c_k = 1$ for one specific $k \in \{1, \dots, n\}$ and zero otherwise, we get the partial derivative of u_l by u_k, but we can compute any other directional derivatives using other vectors $(c_1, \dots, c_n)$.

Next we consider nodes with maximum distance 2 from all nodes $u_1, \dots, u_n$. For such a node u_l

$$c_l = \sum_{i=1,\dots,n} \frac{\partial u_l}{\partial u_i} c_i = \sum_{i=1,\dots,n} \sum_{k|(k,l)\in\Lambda} \frac{\partial u_l}{\partial u_k}\frac{\partial u_k}{\partial u_i} c_i = \sum_{k|(k,l)\in\Lambda} \frac{\partial u_l}{\partial u_k} c_k \tag{2.3}$$

where we can assume that the c_k were computed in the previous step, because their maximum distance to all initial nodes $u_1, \dots, u_n$ muss be less than 2, hence 1. The same argument can be iterated for nodes with maximum distance $3, 4, \dots$ until we reach the final nodes $u_{K-m+1}, \dots, u_K$. This way we can work forward through the computational graph and compute any prescribed directional derivative.

3. Backward mode

We start at the final nodes and compute for all nodes u_l with maximum distance 1 from all of these nodes

$$\overline{c_l} = \sum_{i=K-m+1,\dots,K} \frac{\partial u_i}{\partial u_l}\overline{c_i} \tag{3.1}$$

Note that we compute a weighted sum in the dependent variables now. By setting a specific c_k to 1 and the rest to zero again we can compute the partial derivatives of a single final variable. Again using the chain rule we can compute

(3.2)
$$\overline{c_l} = \sum_{i=K-m+1,\ldots,K} \frac{\partial u_i}{\partial u_l}\overline{c_i} = \sum_{i=K-m+1,\ldots,K} \sum_{k|(l,k)\in\Lambda} \frac{\partial u_i}{\partial u_k}\frac{\partial u_k}{\partial u_l}\overline{c_i} = \sum_{k|(l,k)\in\Lambda} \frac{\partial u_k}{\partial u_l}\overline{c_k}$$

for all nodes u_l with maximum distance of 2 from all the final node. Note that the chain rule formally requires to include all indices k on which u_i depends. Howvever if u_k does not depend on u_l the term will effectively be zero, so we can drop these summands from the beginning. Also we may include indices k on which u_i does not depend in the first place, which is not harmful for the same reason.

As above we can assume all $\overline{c_k}$ to be computed in the previous step, so that we can iterate backwards to the inital nodes to get all partial derivatives of the weighted sum of the final nodes w.r.t. the initial nodes.

References

[1] QuantLib A free/open-source library for quantitative finance, http://www.quantlib.org

E-mail address, P. Caspers: pcaspers1973@googlemail.com