QR and LQ Decomposition Matrix Backpropagation Algorithms for Square, Wide, and Deep - Real or Complex - Matrices and Their Software Implementation

8th International Conference on Algorithmic Differentiation, CHICAGO, ILLINOIS

Lucas Roberts & Denisa Olteanu Roberts

UNIVERSITY OF ILLINOIS CHICAGO

Algorithmic Differentation

- Matrix Background
- Algorithmic Differentiation Background
- Paper Contributions
- Future Ideas

Matrix Background

An {over,re}view of some Matrix Analysis material.

Matrix Concepts I.

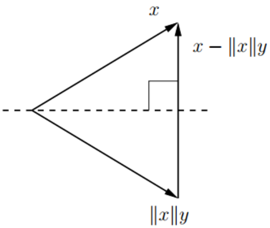
- Linear Independence (LIN) $(x_0, ..., x_{n-1})$: are said to be LIN iff $\sum_{i=0}^{n-1} a_i x_i = 0$ with not all $a_i = 0$.
- Rank of a matrix: The largest number columns that constitute a LIN set of columns of the matrix.
- Partitioning: A matrix may be partitioned into sub-matrices $\boldsymbol{A} = [\boldsymbol{X}|\boldsymbol{Y}]$. Here the $\#\{\text{rows of } \boldsymbol{X}\} + \#\{\text{rows of } \boldsymbol{Y}\} = \#\{\text{rows of } \boldsymbol{A}\}$ and *mutatis mutandis* for columns.

Matrix Concepts II.

- Householder Reflection: Let $v \in \mathbb{R}^n$ be a non-zero vector. A Householder vector of v is of the form $\boldsymbol{P} = \boldsymbol{I} - 2\frac{vv^T}{v^T v}$.
- Householder Matrix: $\boldsymbol{Q}$ = $\prod_{i=0}^{n-1} \boldsymbol{Q}_i$, a product of n distinct Householder vectors.
- QR Factorization: $\boldsymbol{A}_{n\times m} = \boldsymbol{Q}_{n\times m}\boldsymbol{R}_{m\times m}$ where $n \geq m$.

Some Intuition

- Given a vector x we want to find a reflection that transforms x into a direction parallel to some unit vector y.
- To achieve this we do $u = x - ||x||y$ and $v = u/||u||$.
- Then calculate $\boldsymbol{I} - 2vv^T = ||x||y$.
- Now by choosing $y = e_1$, the Euclidean basis vector with first element 1, we recurse on corresponding vectors to arrive at an upper diagonal matrix $\boldsymbol{R}$.
- $\boldsymbol{Q}$ is the product of the Householder reflection matrices and $\boldsymbol{A} = \boldsymbol{QR}$.



Intuition (cont)

- Start with the first column of A and zero out all entries beneath the main diagonal entry.
- Then recurse onto the $(n-1) \times (m-1)$ sub-matrix. Repeat until empty matrix.
- This gives you $\boldsymbol{Q}^T\boldsymbol{A} = \boldsymbol{R}$.

$$
\underset{A}{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}}
\xrightarrow{Q_1}
\underset{Q_1A}{\begin{bmatrix} \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}
\xrightarrow{Q_2}
\underset{Q_2Q_1A}{\begin{bmatrix} \times & \times & \times \\ & \boldsymbol{\times} & \boldsymbol{\times} \\ & \mathbf{0} & \boldsymbol{\times} \\ & \mathbf{0} & \boldsymbol{\times} \\ & \mathbf{0} & \boldsymbol{\times} \end{bmatrix}}
\xrightarrow{Q_3}
\underset{Q_3Q_2Q_1A}{\begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & & \boldsymbol{\times} \\ & & \mathbf{0} \\ & & \mathbf{0} \end{bmatrix}}
$$

Useful QR Facts

- $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}$.
- $\boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{I}_{n \times n}$ and $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}_{m \times m}$.
- $\text{span}(\boldsymbol{A}) = \text{span}(\boldsymbol{Q})$.

Span is the collection of all vectors which can be represented as linear combinations of the basis from the vector space.

Algorithmic Differentiation Background

An {over,re}view of some Algorithmic Differentiation material.

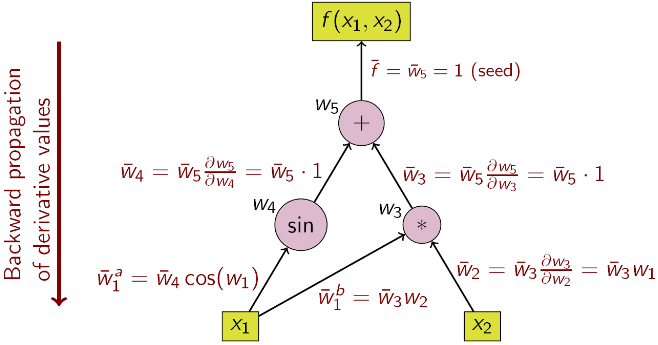
Brief Algorithmic Differentiation Intro I

- Reverse mode will be the primary focus of the talk.
- In algo-diff, represent a function as a Directed Acyclic Graph (DAG).
- Then topologically sort the graph-if necessary.
- Then generate intermediate node values on the forward pass.
- In the reverse pass generate the gradient values.

Brief Algorithmic Differentiation Intro II

- In many examples, the node values are expressed as scalars. However, the same approach applies if the node values are matrices.
- As we'll see, the equations become more difficult to derive. However, the closed form solutions enable speedups.
- A motivating idea for matrix algorithmic differentiation is to use the BLAS routines directly when possible, see Seeger et al. also Giles paper.

Brief Algorithmic Differentiation Example



$\bar{x}_1 = \bar{w}_1^a + \bar{w}_1^b = \cos(x_1) + x_2 \qquad \bar{x}_2 = \bar{w}_2 = x_1$

Paper Contributions

E.g. What is new and worth noting and citing.

Equations Non-wide Case

- For $\boldsymbol{A} \in \mathbb{R}^{r\times c}$ let $\boldsymbol{A} = \boldsymbol{QR}$ then,
  $\bar{\boldsymbol{A}} = \left(\bar{\boldsymbol{Q}} + \boldsymbol{Q} copyltu(\boldsymbol{M})\right) \boldsymbol{R}^{-T}$ with $\boldsymbol{M} = \boldsymbol{R}\bar{\boldsymbol{R}}^T - \bar{\boldsymbol{Q}}^T\boldsymbol{Q}$, (eqn 3.3)
- $\bar{\boldsymbol{A}} = \boldsymbol{Q}\left[\bar{\boldsymbol{R}} + \boldsymbol{P}_L \circ \left(\boldsymbol{R}\bar{\boldsymbol{R}}^T - \bar{\boldsymbol{R}}\boldsymbol{R}^T + \boldsymbol{Q}^T\bar{\boldsymbol{Q}} - \bar{\boldsymbol{Q}}^T\boldsymbol{Q}\right)\boldsymbol{R}^{-T}\right] + \left(\bar{\boldsymbol{Q}} - \boldsymbol{Q}\boldsymbol{Q}^T\bar{\boldsymbol{Q}}\right)\boldsymbol{R}^{-T}$ (eqn 3.8) (prior work from S. Walther)
  $\boldsymbol{P}_L$ is a strictly lower tridiagonal matrix with all ones beneath the diagonal and zeroes along and above the main diagonal
- We will refer to these two equations throughout the remainder of the talk and argue that they are equivalent-or can be made so-and that (3.3) is preferred to (3.8).

Equations Wide Case

- Let $\bar{\boldsymbol{Q}}_p = \bar{\boldsymbol{Q}} + \boldsymbol{Y}\bar{\boldsymbol{V}}^T$.
- For $\boldsymbol{A} \in \mathbb{R}^{r\times c}$ let $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}$ then $\boldsymbol{A} = [\boldsymbol{X}|\boldsymbol{Y}] = \boldsymbol{Q}\boldsymbol{R} = \boldsymbol{Q}[\boldsymbol{U}|\boldsymbol{V}]$.
- $\bar{\boldsymbol{A}} = [(\bar{\boldsymbol{Q}}_p + \boldsymbol{Q}\mathit{copyltu}(\boldsymbol{M}))\,\boldsymbol{U}^{-T}|\bar{\boldsymbol{Y}}]$, (eqn 3.3).
- $\bar{\boldsymbol{A}} = [\boldsymbol{Q}\left(\bar{\boldsymbol{Q}}_p + \boldsymbol{P}_L \circ \left(\boldsymbol{U}\bar{\boldsymbol{U}}^T - \bar{\boldsymbol{U}}\boldsymbol{U}^T + \boldsymbol{Q}^T\bar{\boldsymbol{Q}}_p - \bar{\boldsymbol{Q}}_p^T\boldsymbol{Q}\right)\boldsymbol{U}^{-T}\right] + \left(\bar{\boldsymbol{Q}}_p - \boldsymbol{Q}\boldsymbol{Q}^T\bar{\boldsymbol{Q}}_p\right)\boldsymbol{U}^{-T}|\bar{\boldsymbol{Y}}]$.
- In both equations $\bar{\boldsymbol{Y}} = \boldsymbol{Q}\bar{\boldsymbol{V}}$.
- If $\boldsymbol{X}$ is not full rank use an rank revealing QR and permute the columns to get the columns first, $\boldsymbol{A}\boldsymbol{P}_\pi = \boldsymbol{Q}\boldsymbol{R}$. For $\boldsymbol{P}_\pi$ the permutation matrix.
- *Define the matrix product of $\boldsymbol{Y}$ or $\bar{\boldsymbol{Y}}$ to generate a $\boldsymbol{0}$ when $\boldsymbol{Y}$, $\boldsymbol{V}$ are empty then these equations give the non-wide equations as a special case.

Paper Contributions

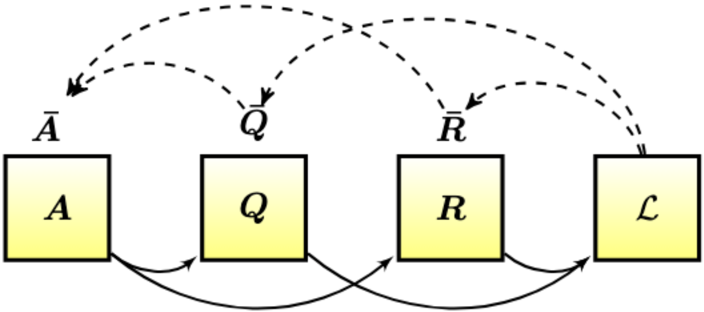
- Proof of Equivalence of Eqn 3.3 vs Eqn 3.8. for $\mathbb{R}$ and $\mathbb{C}$ fields.
- Full proofs of QR derivative formulae from first principles for wide case (rows $<$ columns), for both $\mathbb{R}$ and $\mathbb{C}$.
- Correction term for $\mathbb{C}$ field when using Eqn 3.8.
- Implementations in wide case for major open source deep learning frameworks (PyTorch & Tensorflow), for both $\mathbb{R}$ and $\mathbb{C}$.
- Also tall/deep case of LQ decomposition, analogous to transpose, of QR.

For completeness, also include derivations of gradients for the tall/deep and the square QR cases for $\mathbb{R}$.

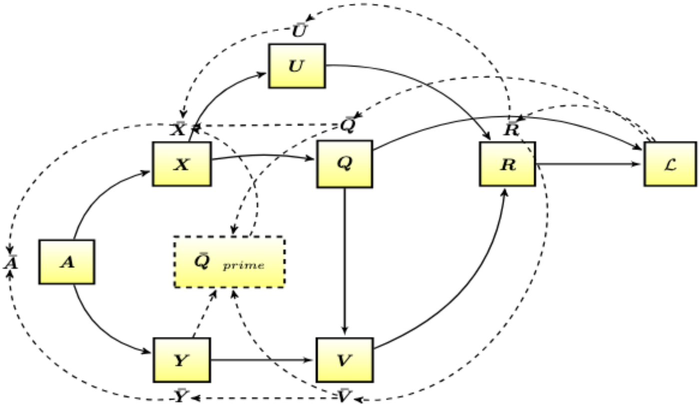
Ideas Driving The Proofs

- $\boldsymbol{C} = f(\boldsymbol{A}, \boldsymbol{B})$ where f is some function with matrix argument(s), for us f is a QR factorization.
- Express loss: $d\mathcal{L} = \operatorname{tr}(\bar{\boldsymbol{C}}^T d\boldsymbol{C}) = \operatorname{tr}(\bar{\boldsymbol{C}}^T \frac{\partial f}{\partial \boldsymbol{A}} d\boldsymbol{A}) + \operatorname{tr}(\bar{\boldsymbol{C}}^T \frac{\partial f}{\partial \boldsymbol{B}} d\boldsymbol{B})$.
- Then, identify $\bar{\boldsymbol{A}} = \frac{\partial f}{\partial \boldsymbol{A}}^T \bar{\boldsymbol{C}}$ and $\bar{\boldsymbol{B}} = \frac{\partial f}{\partial \boldsymbol{B}}^T \bar{\boldsymbol{C}}$, as the variations/gradients sought.
- (Wide case) Partition the input matrix $\boldsymbol{A} = [\boldsymbol{X} | \boldsymbol{Y}]$ with $\boldsymbol{X}$ square and Y tall/deep.
- Assume $\boldsymbol{X}$ is full column rank, then $\boldsymbol{Y}$ can be expressed as a linear combination of $\boldsymbol{X}$.
- Analogous partition of $\boldsymbol{R}$ as $\boldsymbol{R} = [\boldsymbol{U} | \boldsymbol{V}]$.

QR Algo-diff Data Flow Square/Deep-Tall



QR Algo-diff Data Flow Wide Case



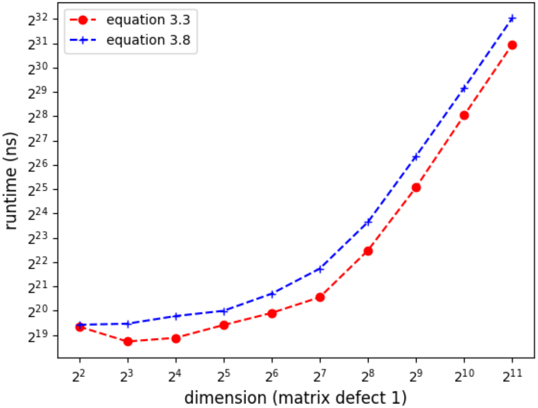
Key Point-A New Matrix

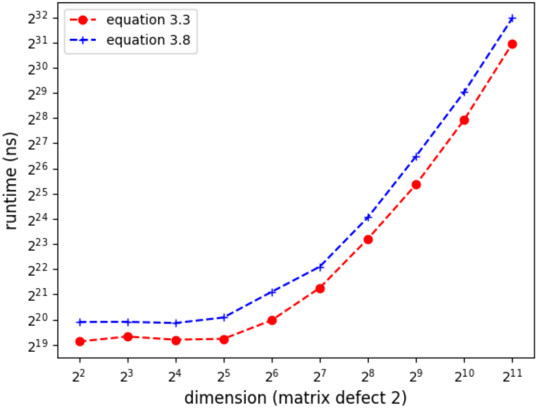
- We define a new gradient matrix $\bar{\boldsymbol{Q}}_p$, denoted $\bar{\boldsymbol{Q}}_{prime}$ in our paper.
- In deep/square cases this corresponds to the original $\bar{\boldsymbol{Q}}$ (e.g. a special case).
- Allows use of Equation (3.8) if desired for wide $\boldsymbol{A}$ matrices by replacing all instances of $\bar{\boldsymbol{Q}}$ with $\bar{\boldsymbol{Q}}_p$ in gradient.
- However, we prefer equation 3.3.

Simulated Matrices Setup-Varying Matrix Defect

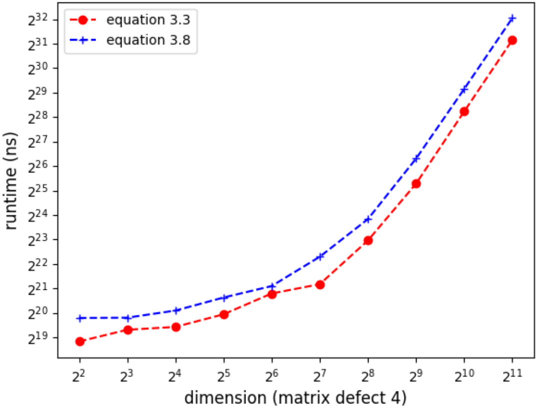
- We call the matrix defect the value by which we multiply rows to get columns.
- For example, with a defect of 2, we have 4 rows, 8 columns.
- A square matrix has defect 1.
- All matrix entries are Gaussian mean 0, variance 1.

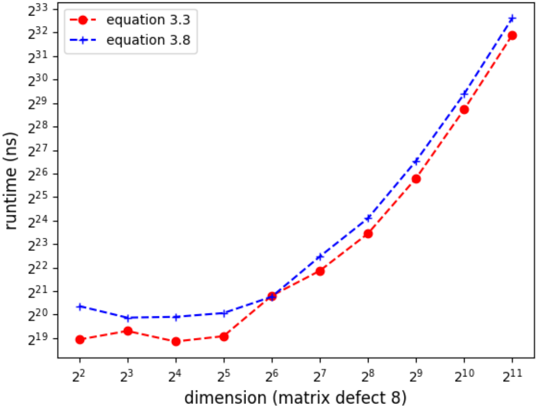
Equation 3.3 vs 3.8 Runtimes





Equation Runtimes-Larger Defects





Equation 3.3 vs 3.8 Matrix Market Examples

- The simulated values indicate some form of expected behavior. What if naturally occuring matrices are not what we expect?
- Let's look at a few matrix market examples-Harwell Boeing economic matrices.
- In all 3 cases the use of equation 3.3 has a better runtime.

Matrix Market Examples-All Times In (ns)

From The Harwell-Boeing Economic Models

Matrix	Eqn 3.3	Eqn 3.8	rows	columns
Wm1	4553340	11666639	207	277
Wm2	3677850	8686058	207	260
Wm3	3465780	7900550	207	260

Equation 3.3 vs 3.8 Runtimes

- Prefer Equation 3.3, why? The evidence:
- The TLDR (too long, didn't read) here is that using equation 3.3 is faster on average than equation 3.8.
- For larger defects, the difference decreases. This is because the proportion of the total computation being done is by the calculation required for the additional $(columns - rows)$ columns, e.g. the wide/defect part. Even here Equation 3.3 is modestly faster.
- Using Equation 3.3 obviates using a special branch for $\mathbb{C}$ fields, simpler code and easier to maintain and also faster.

Why $\mathbb{C}$ Is Different?

- The reason falls out naturally from the proof in the wide case.
- For the proof to work for $\mathbb{C}$ we need $\boldsymbol{P}_L \circ \left(\boldsymbol{M} - \boldsymbol{M}^\dagger\right) + \boldsymbol{M}^\dagger$ to equal $\text{symh}(\boldsymbol{M} \circ \boldsymbol{E})$ where $\boldsymbol{E}$ is a matrix of 1s along the main diagonal, 0s above and 2s beneath. For complex, wide matrices these two are not equal.
- We derive a correction term to make them equal, $\mathcal{C} = i\Im(\text{diag}(\boldsymbol{M}))$, where $i = \sqrt{-1}$.
- Use the correction term if you want to use Equation 3.8 with complex valued matrices.

Some Future Research Areas

Get in touch if interested

Immediate Next Steps: JAX?

- I plan to implement the wide case in JAX if the maintainers will accept a PR. I've initiated discussion on a github issue. If you are someone who maintains or contributes to JAX-or who knows the codebase well-please let's talk during the conference.
- Julia? I'm not a Julia developer. If a Julia developer wants to contribute this-and they do not already support I'm happy to work with you to help ensure the code is correctly implemented.

References Cited

- Algorithmic Differentiation of Linear Algebra Functions with Application in Optimum Experimental Design, S. Walther and L. Lehmann
- An extended collection of matrix derivative results for forward and reverse mode algorithmic differentiation, Mike Giles
- Auto-Differentiating Linear Algebra, Seeger et al.
- Fast Differentiable Sorting and Ranking, Blondel et al. Gini-regularized Optimal Transport with an Application to Spatio-Temporal Forecasting, Roberts et al. Neurips 2017 Smart Vision-Language Reasoners, Roberts and Roberts ICML 2024

Some (Natural) Extensions

A non-exhaustive list.

- Parallelization
- Special matrix structures
- Partitioning technique for other matrix factorizations
- Extend to rank revealing QR via propagating *an approximate* gradient through the (learned) permutation, $\boldsymbol{P}_\pi$ to determine $\bar{\boldsymbol{P}}_\pi$.

Other Research Areas Of Interest

Other things you to speak to me about during coffee breaks

- NLP and IR
- Multimodal LLMs
- Math AI: Smarter VLMs
  - — Using images and fine tuning (ICML 2024)
  - — VLMs-(submitted Neurips 2024)
  - — Scaling laws/data paucity
- AI generated content detection (WIP)

Thank you!