Differentiating sparse matrix operations with reversible programming

Jie Li

Mentors: Jinguo Liu, Jiuning Chen

September 29, 2021

1 Project Information

1.1 Scheme Description

Sparse matrices are extensively used in scientific computing, however there is no automatic differentiation package in Julia yet to handle sparse matrix operations. This project utilizes the reversible embedded domain-specific language NiLang.jl to differentiate sparse matrix operations by writing the sparse matrix operations in a reversible style. The generated backward rules are ported to ChainRules.jl as an extension, so that one can access these features in an automatic differentiation package like Zygote, Flux and Diffractor directly.

1.2 Time Planning

This project is shipped by four (mostly) sequential stages:

(1) Implement low level operations by NiLang.

(2) Carefully test by CI and export chain rules into ChainRules.jl.

(3) Implement high level operations and perform AD operations.

(4) Add some use cases and enhance docs.

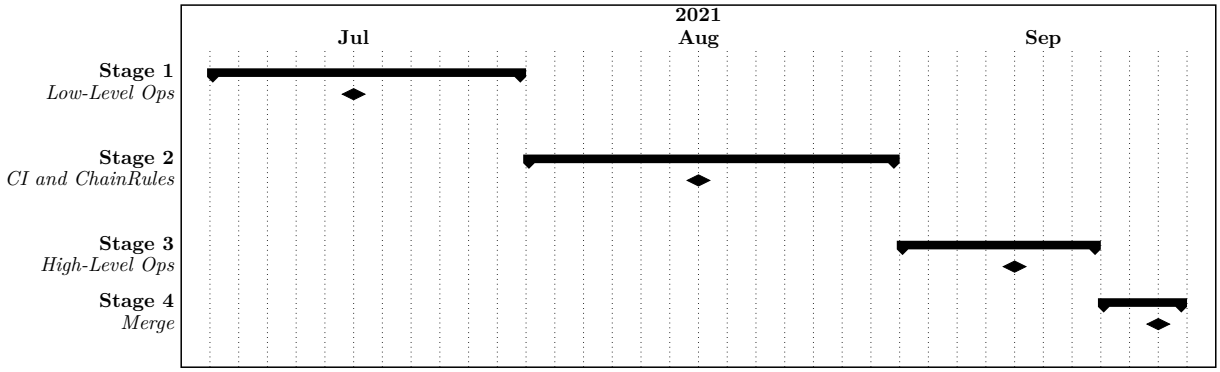
I list the timeline of this project in the form of Gantt chart.

(1) 1st Jul - 31st Jul Implement low level operations by NiLang.

(2) 1st Aug - 15th Aug Carefully test by CI and export chain rules into ChainRules.jl.

(3) 15th Sep - 20th Sep Implement high level operations and perform AD operations.

(4) 21st Sep - 30th Sep Add some use cases and enhance docs.



2 Project Summary

2.1 Project Output

(1) Differentiate sparse matrix operations in Julia base by rewriting the sparse functions in NiLang.jl.
Status: *Already completed. Black-box tests on NiLang programs have passed in CI.* Related PR

(2) Port the generated backward rules to ChainRules.jl as an extension.
Status: *Already completed. Backward rules have been checked by ChainRulesTestUtils.* Related PR

(3) Release an open source julia package with test coverage over 85%.
Status: *Test coverage achieved at 87%.* Code Converage Report

(4) Add some use cases for getting a start.
Status: *Already completed. A simple use case is shown in ReadMe.* Related PR

2.2 Scheme Progress

1. 1st Jul - 31st Jul Implement low level operations by NiLang.
   I have rewritten almost all sparse matrix multiplication and dot operations in Julia base by NiLang. I list the implemented sparse matrix operators as follows. All the functions listed have passed CI tests and the code coverage have achieved over 85%.

```
1        function imul!(C::StridedVecOrMat, A::AbstractSparseMatrix{T}, B::DenseInputVecOrMat, ::
        Number, ::Number) where T
2        function imul!(C::StridedVecOrMat, xA::Adjoint{T, <:AbstractSparseMatrix}, B::
        DenseInputVecOrMat, ::Number, ::Number) where T
3        function imul!(C::StridedVecOrMat, X::DenseMatrixUnion, A::AbstractSparseMatrix{T}, ::Number,
         ::Number) where T
4        function imul!(C::StridedVecOrMat, X::Adjoint{T1, <:DenseMatrixUnion}, A::
        AbstractSparseMatrix{T2}, ::Number, ::Number) where {T1, T2}
5        function imul!(C::StridedVecOrMat, X::DenseMatrixUnion, xA::Adjoint{T, <:AbstractSparseMatrix
        }, ::Number, ::Number) where T
6        function idot(r, A::SparseMatrixCSC{T},B::SparseMatrixCSC{T}) where {T}
7        function idot(r, x::AbstractVector, A::AbstractSparseMatrix{T1}, y::AbstractVector{T2}) where {
        T1, T2}
8        function idot(r, x::SparseVector, A::AbstractSparseMatrix{T1}, y::SparseVector{T2}) where {T1,
        T2}
9
```

2. 1st Aug - 15th Aug Carefully test by CI and export chain rules into ChainRules.jl.
   I have exported the backward rules into ChainRules.jl. I used ChainRulesTestUtils.jl to test correctness and robustness of sparse matrix AD rules. All the rules have passed CI tests.

3. 15th Sep - 20th Sep Implement high level operations and perform AD operation.
   I have implemented low rank svd algorithm in Julia. To perform AD on low_rank_svd algorithm, I wrapped backward rules of QR decomposition since QR rules haven't been ensembled in ChainRules. I used finite differences method to check the correctness of the algorithm. All the code have passed CI tests and the code converage have achieved at 87% finally.

4. 21st Sep - 30th Sep Add some use cases and enhance docs.
   A simple use case has been added into ReadMe in the project.

2.3 Problems and Solutions

You could focus on the summary and experience

2.4 Development Quality

You could give a self evaluation of development quality

2.5 Communication and Feedback with Mentor