Package ‘einsum’

May 15, 2021

Type Package

Title Einstein Summation

Version 0.1.0

Description
The summation notation suggested by Einstein (1916) <doi:10.1002/andp.19163540702> is a concise mathematical notation that implicitly sums over repeated indices of n-dimensional arrays. Many ordinary
matrix operations (e.g. transpose, matrix multiplication, scalar product, 'diag()', trace etc.) can be written using Einstein notation. The notation is particularly convenient for expressing operations on arrays with more than two dimensions because the respective operators ('tensor products') might not have a standardized name.

License MIT + file LICENSE

Encoding UTF-8

SystemRequirements C++11

Suggests testthat,
covr

RdMacros mathjaxr

RoxygenNote 7.1.1

LinkingTo Rcpp

Imports Rcpp,
glue,
mathjaxr

R topics documented:

einsum *Einstein Summation*

Description

Einstein summation is a convenient and concise notation for operations on n-dimensional arrays.

Usage

```
einsum(equation_string, ...)

einsum_generator(equation_string, compile_function = TRUE)
```

Arguments

`equation_string`
: a string in Einstein notation where arrays are separated by ',' and the result is separated by '->'. For example `"ij,jk->ik"` corresponds to a standard matrix multiplication. Whitespace inside the `equation_string` is ignored. Unlike the equivalent functions in Python, `einsum()` only supports the explicit mode. This means that the `equation_string` must contain '->'.

`...`
: the arrays that are combined. All arguments are converted to arrays with `as.array`.

`compile_function`
: boolean that decides if `einsum_generator()` returns the result of `Rcpp::cppFunction()` or the program as a string. Default: `TRUE`.

Details

The following table show, how the Einstein notation abbreviates complex summation for arrays/matrices:

equation_string	Formula	
`"ij,jk->ik"`	$Y_{ik} = \sum_j A_{ij} B_{jk}$	Matrix multiplication
`"ij->ji"`‘	$Y = A^T$	Transpose
`"ii->i"`	$y = \text{diag}(A)$	Diagonal
`"ii->ii"`	$Y = \text{diag}(A) I$	Diagonal times Identity
`"ii->"`	$y = \text{trace}(A) = \sum_i A_{ii}$	Trace
`"ijk,mjj->i"`	$y_i = \sum_j \sum_k \sum_m A_{ijk} B_{mjj}$	Complex 3D operation

The function and the conventions are inspired by the `einsum()` function in NumPy (documentation). Unlike NumPy, 'einsum' only supports the explicit mode. The explicit mode is more flexible and can avoid confusion. The common summary of the Einstein summation to "sum over duplicated indices" however is not a good mental model. A better rule of thumb is "sum over all indices not in the result".

Note: `einsum()` internally uses C++ code to provide results quickly, the repeated parsing of the `equation_string` comes with some overhead. Thus, if you need to do the same calculation over and over again it can be worth to use `einsum_generator()` and call the returned the function. `einsum_generator()` generates efficient C++ code that can be one or two orders of magnitude faster than `einsum()`.

Value

The einsum() function returns an array with one dimension for each index in the result of the equation_string. For example "ij,jk->ik" produces a 2-dimensional array, "abc,cd,de->abe" produces a 3-dimensional array.

The einsum_generator() function returns a function that takes one array for each comma-separated input in the equation_string and returns the same result as einsum(). Or if compile_function = FALSE, einsum_generator() function returns a string with the C++ code for such a function.

Examples

```
mat1 <- matrix(rnorm(n = 4 * 8), nrow = 4, ncol = 8)
mat2 <- matrix(rnorm(n = 8 * 3), nrow = 8, ncol = 3)

# Matrix Multiply
mat1 %*% mat2
einsum("ij,jk -> ik", mat1, mat2)

# einsum_generator() works just like einsum() but returns a performant function
mat_mult <- einsum_generator("ij,jk -> ik")
mat_mult(mat1, mat2)

# Diag
mat_sq <- matrix(rnorm(n = 4 * 4), nrow = 4, ncol = 4)
diag(mat_sq)
einsum("ii->i", mat_sq)
einsum("ii->ii", mat_sq)

# Trace
sum(diag(mat_sq))
einsum("ii->", mat_sq)


# Scalar product
mat3 <- matrix(rnorm(n = 4 * 8), nrow = 4, ncol = 8)
mat3 * mat1
einsum("ij,ij->ij", mat3, mat1)

# Transpose
t(mat1)
einsum("ij->ji", mat1)


# Batched L2 norm
arr1 <- array(c(mat1, mat3), dim = c(dim(mat1), 2))
c(sum(mat1^2), sum(mat3^2))
einsum("ijb,ijb->b", arr1, arr1)
```

---

einsum_package *Einstein Summation*

Description

Einstein summation is a concise mathematical notation that implicitly sums over repeated indices of n-dimensional arrays. Many ordinary matrix operations (e.g. transpose, matrix multiplication, scalar product, 'diag()', trace etc.) can be written using Einstein notation. The notation is particularly convenient for expressing operations on arrays with more than two dimensions because the respective operators ('tensor products') might not have a standardized name.

Index