Package: armacmp (via r-universe)

July 7, 2024

Title Translate R Linear Algebra Code to Armadillo C++

Version 0.1.0.9000

Description Compile linear algebra R code to C++ using the Armadillo Template Library. The package further supports mathematical optimization purely in C++.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Imports Rcpp, RcppArmadillo, RcppEnsmallen, methods, R6, utils, stats

Suggests testthat (>= 2.1.0), covr, knitr, rmarkdown

SystemRequirements C++11

URL https://github.com/dirkschumacher/armacmp

BugReports https://github.com/dirkschumacher/armacmp/issues

VignetteBuilder knitr

Collate 'ast-classes.R' 'annotate-ast.R' 'armacmp-package.R' 'armacmp.R' 'compiler.R' 'optimizers.R' 'optim.R' 'types.R'

Repository https://fastverse.r-universe.dev

RemoteUrl https://github.com/dirkschumacher/armacmp

RemoteRef HEAD

RemoteSha 8dad8c9d71bae144c7ec250c83cef72654587305

Contents

`compile` *Compile Linear Algebra Code to C++*

Description

Compile Linear Algebra Code to C++

Usage

```
compile(fun, verbose = FALSE)
```

Arguments

`fun`	a function
`verbose`	optional logical, print out compiler information

This function always compiles functions. Every function needs to have a `return` statement with an optional type argument. All input parameters are by default of type double matrix. Type inference is tried to be done, but sometimes it is helpful to add type annotation.

Take a look at function reference vignette for more information.

Examples

```
## Not run:
trans <- compile(function(X) {
  return(t(X))
})
trans(matrix(1:10))

## End(Not run)
```

`compile_optimization_problem` *Optimize arbitrary and differentiable functions*

Description

The function compiles the code to C++ and uses Armadillo and ensmallen to optimize it.

Usage

```
compile_optimization_problem(
  data = list(),
  evaluate,
  gradient,
  optimizer = optimizer_SA()
)
```

Arguments

`data`	a named list of prior data you would like to supply to the evaluate function.
`evaluate`	a function that is to be minimized. It should return a single numeric.
`gradient`	optional, a function computing the gradient of `evaluate`
`optimizer`	one of the many optimizers

Examples

```
## Not run:
optimize <- compile_optimization_problem(
  data = list(),
  evaluate = function(x) {
    return(2 * norm(x)^2)
  },
  optimizer = optimizer_SA()
)

# should be roughly c(0, 0, 0)
result <- optimize(matrix(c(1, -1, 1), ncol = 1))

## End(Not run)
```

optimizer_CNE *Conventional Neural Evolution Optimizer*

Description

Conventional Neural Evolution Optimizer

Usage

```
optimizer_CNE(
  populationSize = 500,
  maxGenerations = 5000,
  mutationProb = 0.1,
  mutationSize = 0.02,
  selectPercent = 0.2,
  tolerance = 1e-05
)
```

Arguments

populationSize	The number of candidates in the population. This should be at least 4 in size 500
maxGenerations	The maximum number of generations allowed for CNE 5000
mutationProb	Probability that a weight will get mutated 0.1
mutationSize	The range of mutation noise to be added. This range is between 0 and mutation-Size 0.02
selectPercent	The percentage of candidates to select to become the the next generation 0.2
tolerance	The final value of the objective function for termination. If set to negative value, tolerance is not considered 1e-5

optimizer_GradientDescent *Gradient Descent Optimizer*

Description

Gradient Descent Optimizer

Usage

```
optimizer_GradientDescent(
  stepSize = 0.01,
  maxIterations = 1e+05,
  tolerance = 1e-05
)
```

Arguments

stepSize	Step size for each iteration
maxIterations	Maximum number of iterations allowed (0 means no limit).
tolerance	Maximum absolute tolerance to terminate algorithm.

---

optimizer_L_BFGS *L-BFGS Optimizer*

Description

L-BFGS Optimizer

Usage

```
optimizer_L_BFGS(
  numBasis = 10,
  maxIterations = 10000,
  armijoConstant = 1e-04,
  wolfe = 0.9,
  minGradientNorm = 1e-06,
  factr = 1e-15,
  maxLineSearchTrials = 50,
  minStep = 1e-20,
  maxStep = 1e+20
)
```

Arguments

numBasis	Number of memory points to be stored (default 10)
maxIterations	Maximum number of iterations for the optimization (0 means no limit and may run indefinitely)
armijoConstant	Controls the accuracy of the line search routine for determining the Armijo condition
wolfe	Parameter for detecting the Wolfe condition
minGradientNorm	Minimum gradient norm required to continue the optimization
factr	Minimum relative function value decrease to continue the optimization
maxLineSearchTrials	The maximum number of trials for the line search (before giving up)
minStep	The minimum step of the line search
maxStep	The maximum step of the line search

optimizer_SA *Simulated-Annealing with exponential schedule*

Description

Simulated-Annealing with exponential schedule

Usage

```
optimizer_SA()
```

optimizer_SPSA *Simultaneous Perturbation Stochastic Approximation (SPSA)*

Description

Simultaneous Perturbation Stochastic Approximation (SPSA)

Usage

```
optimizer_SPSA(
  alpha = 0.602,
  gamma = 0.101,
  stepSize = 0.16,
  evaluationStepSize = 0.3,
  maxIterations = 1e+05,
  tolerance = 1e-05
)
```

Arguments

alpha	Scaling exponent for the step size.
gamma	Scaling exponent for evaluation step size.
stepSize	Scaling parameter for step size.
evaluationStepSize	Scaling parameter for evaluation step size.
maxIterations	Maximum number of iterations allowed (0 means no limit).
tolerance	Maximum absolute tolerance to terminate algorithm.

translate *Compile a function to C++*

Description

Compile a function to C++

Usage

```
translate(fun, function_name)
```

Arguments

fun	a function
function_name	the function name

Value

a list of type "armacmp_cpp_fun"

type_colvec *Type colvec*

Description

Type colvec

Usage

```
type_colvec()
```

type_matrix *Type matrix*

Description

Type matrix

Usage

```
type_matrix()
```

type_rowvec *Type rowvec*

Description

Type rowvec

Usage

```
type_rowvec()
```

type_scalar_integer *Type int*

Description

Type int

Usage

```
type_scalar_integer()
```

type_scalar_logical *Type logical*

Description

Type logical

Usage

```
type_scalar_logical()
```

type_scalar_numeric *Type numeric*

Description

Type numeric

Usage

```
type_scalar_numeric()
```

Index