

Package ‘splitfngr’

September 16, 2016

Type Package

Title Combined Evaluation and Split Access of Functions

Version 0.1.1

Author Collin Erickson

Maintainer Collin Erickson <collinberickson@gmail.com>

Description Some R functions, such as `optim()`, require a function its gradient passed as separate arguments. When these are expensive to calculate it may be much faster to calculate the function (`fn`) and gradient (`gr`) together since they often share many calculations (chain rule). This package allows the user to pass in a single function that returns both the function and gradient, then splits (hence ‘splitfngr’) them so the results can be accessed separately. The functions provided allow this to be done with any number of functions/values, not just for functions and gradients.

License GPL-3

LazyData TRUE

RoxygenNote 5.0.1

Imports lbfgs

NeedsCompilation no

Repository CRAN

Date/Publication 2016-09-16 18:15:54

R topics documented:

<code>fngr</code>	2
<code>grad_share</code>	3
<code>lbfgs_share</code>	4
<code>make_share</code>	4
<code>optim_share</code>	5

Index	6
--------------	----------

fngr	<i>Access a list of values separately but calculate them together. This function generalizes grad_share for any number of functions.</i>
------	--

Description

Access a list of values separately but calculate them together. This function generalizes grad_share for any number of functions.

Usage

```
fngr(func, evalForNewX = TRUE, recalculate_indices = c(),
      check_all = FALSE)
```

Arguments

func	Function that returns a list of values
evalForNewX	Should the function reevaluate for any new x? Recommended.
recalculate_indices	Indices for which the values should be recalculated. Ignored if evalForNewX is true. Use this if you don't want to pass x to dependent functions, or if you know other indices won't need to be recalculated.
check_all	Should it check that the accessed values were calculated at the current input? Ignored if evalForNewX is true. Will give a warning but still return the stored value.

Value

An environment where the function values are calculated.

Examples

```
tfunc <- function(x) {list(x+1, x+2, x+3, x+4, x+5)}
f <- fngr(tfunc)
f(1)(0)
f(3)(0)
f(3)(1)
f(1)(23.4)
f(4)()

# Use same function but only recalculate when first value is called
g <- fngr(tfunc, evalForNewX = FALSE, recalculate_indices = c(1))
g1 <- g(1)
g3 <- g(3)
g1(1)
g3(1)
g3(11) # This won't be give expected value
g1(11) # This updates all values
g3(11) # This is right
```

grad_share	<i>Calculate function and gradient together but access separately. Reduces computation since they share data in calculation. Doesn't have to be function and gradient, can be any two values calculated together but accessed separately. Useful in optimization when function evaluation is expensive since the chain rule means many parts of function and gradient are the same.</i>
------------	---

Description

Calculate function and gradient together but access separately. Reduces computation since they share data in calculation. Doesn't have to be function and gradient, can be any two values calculated together but accessed separately. Useful in optimization when function evaluation is expensive since the chain rule means many parts of function and gradient are the same.

Usage

```
grad_share(fn_gr)
```

Arguments

fn_gr	A function that returns a list of two values. Both are calculated when fn is called, but only the first is returned. The second is returned when gr is called but nothing is recalculated.
-------	--

Value

An environment with two functions, fn and gr.

Examples

```
quad_share <- function(x){list(sum(x^4), 4*x^3)}  
share <- grad_share(quad_share)  
share$fn(1)  
share$gr(1)  
share$gr(2)  
share$fn(2)  
share$gr(2)
```

lbfgs_share	<i>Use lbfgs packages's lbfgs function but pass in a single function that returns both the function and gradient together in a list. Useful when the function and gradient are expensive to calculate and can be calculated faster together than separate.</i>
-------------	--

Description

Use lbfgs packages's lbfgs function but pass in a single function that returns both the function and gradient together in a list. Useful when the function and gradient are expensive to calculate and can be calculated faster together than separate.

Usage

```
lbfgs_share(fngr, vars, ...)
```

Arguments

fngr	A function that returns a list of two elements: the function value and the gradient value.
vars	Initial values for the parameters to be optimized over. Will be passed to optim as par argument.
...	Other arguments passed to lbfgs

Value

Result from running lbfgs on the given function

Examples

```
quad_share <- function(x){list(sum(x^4), 4*x^3)}
lbfgs_share(vars=c(3, -5), fngr=quad_share)
```

make_share	<i>Convert a function from multiple function arguments to a single function</i>
------------	---

Description

Convert a function from multiple function arguments to a single function

Usage

```
make_share(func, arg_fn, arg_gr)
```

Arguments

func	The function that takes in two function arguments
arg_fn	The function (first) argument name of func
arg_gr	The gradient (second) argument name of func

Value

A new function that evaluates the two arguments together

Examples

```
quad_share <- function(x){list(sum(x^4), 4*x^3)}
lbfgs_share <- make_share(lbfgs::lbfgs, 'call_eval', 'call_grad')
make_share(lbfgs::lbfgs, 'call_eval', 'call_grad')(quad_share, vars=c(5,-4))
```

optim_share	<i>Use R's optim function but pass in a single function that returns both the function and gradient together in a list. Useful when the function and gradient are expensive to calculate and can be calculated faster together than separate.</i>
-------------	---

Description

Use R's optim function but pass in a single function that returns both the function and gradient together in a list. Useful when the function and gradient are expensive to calculate and can be calculated faster together than separate.

Usage

```
optim_share(par, fngr, ...)
```

Arguments

par	Initial values for the parameters to be optimized over. Will be passed to optim as par argument.
fngr	A function that returns a list of two elements: the function value and the gradient value.
...	Arguments passed to optim, such as method, lower, upper, control, and hessian.

Value

Results from running optim

Examples

```
quad_share <- function(x){list(sum(x^4), 4*x^3)}
optim_share(par=c(3, -5), quad_share, method="BFGS")
```

Index

fngr, [2](#)

grad_share, [3](#)

lbfgs_share, [4](#)

make_share, [4](#)

optim_share, [5](#)