Package: MMAD (via r-universe)

May 26, 2026

Title Minorization-Maximization via Assembly-Decomposition Technology

Version 3.0.0

Description A formula-driven framework for maximizing target functions via the minorization-maximization (MM) algorithm. The package represents the target as a symbolic expression tree, infers its curvature via disciplined-convex-programming rules, and constructs a separable surrogate at each iterate using only Jensen's inequality and the supporting hyperplane. The driver maximizes the surrogate via block-coordinate Newton with line search, falling back to a multivariate step on any non-separable residue. A formula interface accepts standard R expressions (including `sum()` reductions and `X %*% theta` design-matrix products) so statistical models such as Poisson regression can be written in one line.

License GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 2.10)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Repository https://gujq5.r-universe.dev

Date/Publication 2026-05-26 15:13:24 UTC

RemoteUrl https://github.com/gujq5/mmad

RemoteRef HEAD

RemoteSha 7a57a1be1dc89f35eaf6448417f35b93d2206ed6

Contents

`as_mmad_expr` *Convert a formula or expression into an* `mmad_expr`

Description

Convert a formula or expression into an `mmad_expr`

Usage

```
as_mmad_expr(x, init = NULL, data = NULL)
```

Arguments

`x`	A one-sided formula (`~ ...`) or an existing `mmad_expr`.
`init`	Numeric vector of initial parameter values. If named, the names supply the parameter vocabulary; otherwise parameters must be referenced via `theta[i]`.
`data`	Optional list / data frame in which to evaluate theta-free sub-expressions. Default: the formula's environment.

Value

An `mmad_expr`.

Examples

```
expr <- as_mmad_expr(~ log(theta[1] + theta[2]), init = c(0, 0))
evaluate_expr(expr, c(1, 2))$value
```

curvature *Inferred curvature of an* mmad_expr

Description

Returns the DCP curvature of the expression as one of "affine", "convex", "concave", or "unknown". "unknown" means the DCP rule set could not prove a definite curvature; this is a refusal, not a claim of non-convexity.

Usage

```
curvature(expr)
```

Arguments

expr	An mmad_expr.

Value

A length-1 character.

Examples

```
curvature(log(mmad_var(1) + 1))                    # "concave"
curvature(exp(mmad_var(1)))                        # "convex"
curvature(2 * mmad_var(1) + 3 * mmad_var(2))       # "affine"
curvature(exp(mmad_var(1)) - exp(mmad_var(2)))     # "unknown"
```

evaluate_expr *Evaluate an* mmad_expr *at a parameter vector*

Description

Computes the value, gradient, and Hessian of the expression at theta, all with respect to theta. The gradient is a numeric vector of length length(theta); the Hessian is a length(theta) by length(theta) numeric matrix.

Usage

```
evaluate_expr(expr, theta)
```

Arguments

expr	An object of class mmad_expr.
theta	A numeric vector of parameter values.

Details

This is the Phase 1 replacement for `Function_evaluation()` – it works on the new expression tree. The legacy function continues to work on the old list representation; `legacy_to_expr()` bridges between them.

Value

A list with components `value`, `gradient`, and `hessian`.

Examples

```
expr <- log(mmad_var(1) + mmad_var(2))
evaluate_expr(expr, c(1, 1))
```

`Function_check` *Diagnose an* `mmad_expr` *(or formula) before optimization*

Description

Runs curvature inference, top-level summand decomposition, domain check at `init`, and a dry-run of `minorize_at()`. Returns a structured report with a `print()` method intended as a quick health check before invoking `mmad()`.

Usage

```
Function_check(expr, init, data = NULL, tree = FALSE)
```

Arguments

`expr`	A formula or `mmad_expr` representing the target.
`init`	Numeric vector of initial parameter values (named or not).
`data`	Optional `list`/`data.frame` of theta-free symbols referenced in `expr` (only relevant when `expr` is a formula).
`tree`	Logical; if `TRUE`, `print()` will also render the full annotated expression tree showing the minorizability of every node. Defaults to `FALSE`. The tree data is always built and stored in `$expr_tree` regardless of this flag; the flag only affects printing.

Details

The `is_dcp` flag reflects strict DCP composition rules and may be `FALSE` even for expressions that are fully minorizable by MMAD's extended rules (e.g. `log(1 + exp(theta))` or `exp(log(theta1) + log(theta2))`). The operationally relevant criterion is `is_separable`: if that is `TRUE`, `mmad()` will optimise the expression successfully.

The `print()` method renders a full expression tree in which every node is annotated with `[curvature | sign | minorizabl`

Value

A list with class "mmad_check" and components target_curvature, target_sign, is_dcp, is_minorizable, summands, domain_ok, domain_message, is_separable, non_separable_indices, expr_tree, show_tree.

Examples

```
chk <- Function_check(~ log(theta[1] + theta[2]) - theta[1],
                      init = c(1, 1))
print(chk)
print(chk, tree = TRUE)   # show the expression tree
```

---

is_dcp *Whether an expression's curvature is provably one of affine/convex/concave*

Description

Convenience predicate: TRUE exactly when curvature() is not "unknown".

Usage

```
is_dcp(expr)
```

Arguments

expr	An mmad_expr.

Value

TRUE or FALSE.

Examples

```
is_dcp(log(mmad_var(1) + 1))                      # TRUE
is_dcp(exp(mmad_var(1)) - exp(mmad_var(2)))       # FALSE
```

Math.mmad_expr *Mathematical functions for* mmad_expr

Description

log(), exp(), and sqrt() (the latter as (.)^(1/2)).

Usage

```
## S3 method for class 'mmad_expr'
Math(x, ...)
```

Arguments

x	An mmad_expr.
...	Unused.

Value

An mmad_expr.

minorize_at *Build a surrogate of an* mmad_expr *at a current iterate*

Description

Given a target expr and a current iterate theta_0, construct a surrogate S(theta | theta_0) that lower-bounds expr(theta) and is tangent to it at theta_0. The surrogate is constructed to be as separable across the coordinates of theta as the structure permits; anything that resists clean minorization is preserved as-is in a fallback bucket so that value/gradient/Hessian computations on the full surrogate remain available.

Usage

```
minorize_at(expr, theta_0)
```

Arguments

expr	An mmad_expr.
theta_0	A numeric vector giving the current iterate.

Details

Construction uses only Jensen's inequality and the supporting hyperplane; no per-atom tight surrogates are applied.

Value

A list with components

`expr` the full surrogate as an `mmad_expr`

`constant` the additive constant of the surrogate

`per_coord` a list of length `length(theta_0)` whose `j`-th entry is the 1-d surrogate term in `theta[j]`, or `NULL` if no term touches that coordinate

`non_separable` an `mmad_expr` representing the multi-coordinate residue we could not minorize, or `NULL` if every leaf was minorized or extracted

`is_separable` `TRUE` iff `non_separable` is `NULL`

Examples

```
expr <- log(0.5 * mmad_var(1) + 0.5 * mmad_var(2))
surr <- minorize_at(expr, c(2, 1))
surr$is_separable
evaluate_expr(surr$expr, c(2, 1))$value     # tangent at theta_0
```

`mmad` *Minorization-Maximization driver for an* `mmad_expr` *target*

Description

Given a target expression `expr` and an initial parameter vector `init`, this iteratively maximizes `expr` by constructing a separable surrogate at the current iterate (via `minorize_at()`), taking Newton steps on the separable and entangled blocks, and line-searching on the target to enforce ascent.

Usage

```
mmad(
  expr,
  init,
  data = NULL,
  tol = 1e-06,
  max_iter = 1000,
  line_search_max = 30,
  track_history = FALSE,
  verbose = FALSE
)
```

Arguments

`expr`	An `mmad_expr` or a one-sided formula (`~ ...`). When a formula, it is lowered via `as_mmad_expr()` using `init` and `data` as the parameter vocabulary and data environment.

`init`	Numeric vector of initial parameter values. May be named, in which case the names define the parameter vocabulary (e.g. `init = c(alpha = 1, beta = 0.5)`) and the returned estimate carries those names.
`data`	Optional list / data frame of values for theta-free symbols referenced in the formula. Default: the formula's environment.
`tol`	Convergence tolerance for both the maximum absolute parameter change and the maximum absolute target gradient component.
`max_iter`	Maximum number of MM iterations (default 1000).
`line_search_max`	Maximum number of step-halving steps per iteration's line search (default 30).
`track_history`	If `TRUE`, return a data frame of per-iteration diagnostics (iteration index, value, gradient norm, parameter change). Default `FALSE`.
`verbose`	If `TRUE`, print per-iteration diagnostics.

Value

A list with components

`estimate` the final parameter vector (named if `init` was named)

`value` the target value at `estimate`

`iterations` number of iterations actually run

`converged` `TRUE` iff both convergence criteria were met

`history` per-iteration diagnostics (data frame), or `NULL`

`message` character: short status string

Examples

```
fit <- mmad(~ log(theta[1] + theta[2]) - theta[1] - theta[2] + 2,
            init = c(2, 2))
fit$estimate

# Poisson regression via sum() and X %*% theta:
set.seed(1); n <- 50; p <- 2
X <- matrix(rnorm(n * p, sd = 0.5), nrow = n)
y <- rpois(n, exp(X %*% c(0.3, -0.2)))
mmad(~ sum(y * (X %*% theta) - exp(X %*% theta)),
     init = rep(0, p), data = list(X = X, y = y))
```

mmad_const *Construct a numeric-constant expression node*

Description

Construct a numeric-constant expression node

Usage

```
mmad_const(value)
```

Arguments

value	A finite numeric scalar.

Value

An object of class mmad_expr (subclass mmad_const).

Examples

```
mmad_const(2.5)
```

mmad_var *Construct a parameter reference* theta[i]

Description

Construct a parameter reference theta[i]

Usage

```
mmad_var(i, sign = "unknown")
```

Arguments

i	Positive integer index into the parameter vector.
sign	Optional declaration of the sign of theta[i]. One of "unknown" (default), "positive", "nonneg", "negative", "nonpos", or "zero". The DCP curvature inference (Phase 2) uses this to refine curvature for sign-dependent atoms such as pow.

Value

An object of class mmad_expr (subclass mmad_var).

Examples

```
mmad_var(1)
mmad_var(2, sign = "positive")
```

`Ops.mmad_expr` *Arithmetic operators for* `mmad_expr`

Description

Lets users write expressions naturally: `2 * theta1 - log(theta1 + theta2)` becomes a properly-typed expression tree.

Usage

```
## S3 method for class 'mmad_expr'
Ops(e1, e2)
```

Arguments

`e1`, `e2`	An `mmad_expr` and/or numeric scalar.

Details

Restrictions enforced at construction time:

- `expr * expr` is rejected (one side must be a numeric scalar) – products of two parameter-dependent expressions are non-DCP.
- `numeric / expr` is rejected – division by an expression is non-DCP.
- `expr ^ expr` is rejected – the exponent must be a numeric scalar.

Value

An `mmad_expr`.

sign_of *Inferred sign of an* mmad_expr

Description

Inferred sign of an mmad_expr

Usage

```
sign_of(expr)
```

Arguments

expr	An mmad_expr.

Value

One of "positive", "nonneg", "zero", "nonpos", "negative", "unknown".

Examples

```
sign_of(exp(mmad_var(1)))               # "positive"
sign_of(mmad_var(1) ^ 2)                # "nonneg"
```

Index