

Package: SWORD (via r-universe)

October 29, 2026

Type Package

Title Support Vector Machine Weighted Oblique Random Decision Forests

Version 0.1.0

Description Implements SWORD (Support vector machine Weighted Oblique Random Decision forests), a random forest method for regression based on TORS (Tree Oblique for Regression with weighted SVM) oblique regression trees. At each internal node a Weighted-SVM classifier with a linear kernel defines the oblique splitting hyperplane as a sparse linear combination of predictors. The C++ coordinate-descent solver provides fast training. Variable importance is computed via OIW-VI (Oblique Impurity-Weighted Variable Importance). Provides tree fitting (TORS), forest fitting (SWORD), prediction, and variable importance (VI_SWORD). Parallel tree fitting is supported via the 'future'/'furrr' framework. The SWORD method is described in Carta and Frigau (accepted, in press) 'Oblique Random Forests for Regression via Weighted Support Vector Machine', Advances in Data Analysis and Classification. The TORS algorithm is described in Carta and Frigau (2025) <doi:10.1007/s00180-025-01647-w>. The weighted SVM is based on Yang, Song and Cao (2005) <doi:10.1109/IJCNN.2005.1555965> and the 'WeightSVM' package by Xu et al. (2024) <doi:10.32614/CRAN.package.WeightSVM>.

License GPL-3

URL <https://github.com/acarta88/sword-package>

BugReports <https://github.com/acarta88/sword-package/issues>

Encoding UTF-8

Depends R (>= 4.1.0)

Imports Rcpp (>= 1.0.0)

LinkingTo Rcpp

Suggests infotheo, furrr, progressr, future, visNetwork, microbenchmark, knitr, rmarkdown, ggplot2, MASS, testthat (>= 3.0.0)

VignetteBuilder knitr
Config/roxygen2/version 8.0.0
NeedsCompilation yes
Author Andrea Carta [aut, cre] (ORCID:
 <<https://orcid.org/0000-0003-4077-1822>>), Luca Frigau [aut]
Maintainer Andrea Carta <andrea.cart88@unica.it>
Repository <https://cran.r-universe.dev>
Date/Publication 2026-10-29 16:16:38 UTC
RemoteUrl <https://github.com/cran/SWORD>
RemoteRef HEAD
RemoteSha 1c58e0d9780f866df7e10046f990001fae11a7ed

Contents

pdp_sword	2
plot.sword_flat	4
plot.tors_flat	5
plot_oob_fit	6
plot_tors_text	7
plot_tors_visnet	8
plot_vi_sword	9
predict.sword_flat	10
predict.tors_flat	11
SWORD	11
TORS	14
VI_SWORD	16
Index	18

pdp_sword	<i>Partial dependence plot for a SWORD forest</i>
-----------	---

Description

Computes the marginal effect of a single predictor on forest predictions by fixing the variable to a grid of values and averaging predictions over the full data matrix (Friedman 2001). A rug of observed values is added by default.

Usage

```
pdp_sword(
  forest,
  X,
  var,
  n_grid = 50L,
  rug = TRUE,
  main = NULL,
  xlab = var,
  ylab = "Partial dependence",
  ...
)
```

Arguments

forest	sword_flat object from SWORD .
X	data.frame of predictor values (same columns used to fit the forest; typically the training set).
var	character string; name of the variable to profile.
n_grid	number of equally-spaced grid points over the variable's observed range (default 50).
rug	logical; add a rug of observed values (default TRUE).
main	plot title; NULL generates a default title.
xlab	x-axis label (default: the variable name).
ylab	y-axis label (default "Partial dependence").
...	further graphical arguments passed to plot.

Value

Invisibly returns a data.frame with columns x (grid values, numeric for continuous variables or character for discrete/factor variables) and yhat (partial dependence values).

See Also

[SWORD](#), [plot.sword_flat](#)

Examples

```
set.seed(1)
X <- data.frame(matrix(rnorm(150 * 5), 150, 5,
                      dimnames = list(NULL, paste0("x", 1:5))))
y <- X$x1 * 2 - X$x2 + rnorm(150)
forest <- SWORD(X, y, m = 10, oob = FALSE, verbose = FALSE)
pdp_sword(forest, X, "x1")
```

plot.sword_flat	<i>Plot method for a SWORD forest</i>
-----------------	---------------------------------------

Description

Plot method for a SWORD forest

Usage

```
## S3 method for class 'sword_flat'
plot(x, which = c("oob", "vi"), top_n = 20L, ...)
```

Arguments

x	sword_flat object from SWORD .
which	"oob" for OOB MSE convergence curve (requires oob = TRUE at fit time); "vi" for variable importance barplot.
top_n	maximum variables shown when which = "vi" (default 20).
...	further arguments passed to plot_vi_sword when which = "vi".

Value

Invisibly returns x.

See Also

[plot_vi_sword](#), [plot_oob_fit](#), [pdp_sword](#)

Examples

```
set.seed(1)
X <- data.frame(matrix(rnorm(150 * 5), 150, 5,
                      dimnames = list(NULL, paste0("x", 1:5))))
y <- X$x1 * 2 - X$x2 + rnorm(150)
forest <- SWORD(X, y, m = 10, oob = TRUE, verbose = FALSE)
plot(forest, which = "oob")
plot(forest, which = "vi")
```

plot.tors_flat	<i>Plot a fitted TORS tree</i>
----------------	--------------------------------

Description

Dispatches to [plot_tors_visnet](#) when **visNetwork** is installed; otherwise draws a base-R structure plot (node size proportional to observation count).

Usage

```
## S3 method for class 'tors_flat'
plot(x, use_scaled = TRUE, top_k = 4L, digits = 3L, ...)
```

Arguments

x	tors_flat object from TORS .
use_scaled	logical; use scaled-space coefficients (default TRUE).
top_k	max equation terms shown per node (default 4).
digits	decimal places for coefficients (default 3).
...	ignored.

Value

Invisibly returns x.

See Also

[plot_tors_visnet](#), [plot_tors_text](#)

Examples

```
set.seed(1)
X <- data.frame(matrix(rnorm(150 * 5), 150, 5,
                      dimnames = list(NULL, paste0("x", 1:5))))
y <- X$x1 * 2 - X$x2 + rnorm(150)
tree <- TORS(X, y, nmin = 10, cp = 0.01)
plot(tree)
```

plot_oob_fit	<i>OOB predicted vs observed scatter plot</i>
--------------	---

Description

Plots out-of-bag predictions against the true response values with an identity line and RMSE / R^2 annotation. Requires that the forest was fit with `oob = TRUE`.

Usage

```
plot_oob_fit(forest, y, main = "SWORD - OOB fit", ...)
```

Arguments

forest	sword_flat object fit with <code>oob = TRUE</code> .
y	numeric vector of true response values in the same order as the training data passed to SWORD .
main	plot title (default "SWORD - OOB fit").
...	further graphical arguments passed to plot.

Value

Invisibly returns a list with elements `rmse`, `mae`, and `r2`.

See Also

[SWORD](#), [plot.sword_flat](#)

Examples

```
set.seed(1)
X <- data.frame(matrix(rnorm(150 * 5), 150, 5,
                      dimnames = list(NULL, paste0("x", 1:5))))
y <- X$x1 * 2 - X$x2 + rnorm(150)
forest <- SWORD(X, y, m = 10, oob = TRUE, verbose = FALSE)
plot_oob_fit(forest, y)
```

plot_tors_text	<i>Indented text representation of a TORS tree</i>
----------------	--

Description

Prints the tree node by node with an indented layout. Each internal node shows the hyperplane equation; each leaf shows the predicted mean and sample size.

Usage

```
plot_tors_text(tree, use_scaled = FALSE, top_k = 3L, digits = 3L)
```

Arguments

tree	tors_flat object from TORS .
use_scaled	logical; use scaled-space coefficients (default FALSE).
top_k	maximum number of terms per equation (default 3).
digits	decimal places for coefficients (default 3).

Value

Invisibly returns tree.

See Also

[TORS](#), [plot_tors_visnet](#)

Examples

```
set.seed(1)
X <- data.frame(matrix(rnorm(150 * 5), 150, 5,
                      dimnames = list(NULL, paste0("x", 1:5))))
y <- X$x1 * 2 - X$x2 + rnorm(150)
tree <- TORS(X, y, nmin = 10, cp = 0.01)
plot_tors_text(tree, top_k = 2)
```

plot_tors_visnet	<i>Interactive zoomable visualisation of a TORS tree</i>
------------------	--

Description

Renders an interactive HTML widget of the tree. Supports zoom (mouse wheel), pan (click-drag), and hover tooltips with the full hyperplane equation. Requires the **visNetwork** package.

Usage

```
plot_tors_visnet(tree, use_scaled = TRUE, top_k = 4L, digits = 3L)
```

Arguments

tree	tors_flat object from TORS .
use_scaled	logical; use scaled-space coefficients for labels (default TRUE).
top_k	maximum number of equation terms shown on each node (default 4). Full equation is always in the hover tooltip.
digits	decimal places for coefficients (default 3).

Value

A visNetwork HTML widget (rendered as a side-effect in RStudio / a browser).

See Also

[TORS](#), [plot_tors_text](#)

Examples

```
if (requireNamespace("visNetwork", quietly = TRUE)) {
  set.seed(1)
  X <- data.frame(matrix(rnorm(150 * 5), 150, 5,
                        dimnames = list(NULL, paste0("x", 1:5))))
  y <- X$x1 * 2 - X$x2 + rnorm(150)
  tree <- TORS(X, y, nmin = 10, cp = 0.01)
  plot_tors_visnet(tree)
}
```

plot_vi_sword	<i>Variable importance barplot for a SWORD forest</i>
---------------	---

Description

Calls [VI_SWORD](#) and draws a horizontal barplot of deviance-reduction importance scores, sorted descending.

Usage

```
plot_vi_sword(
  forest,
  top_n = 20L,
  col = "#3498db",
  raw = FALSE,
  main = NULL,
  ...
)
```

Arguments

forest	sword_flat object from SWORD .
top_n	maximum number of variables to display (default 20).
col	bar fill colour (default "#3498db").
raw	logical; if TRUE show importances at the dummy-column level instead of the original variable level (default FALSE).
main	plot title; NULL generates a default title.
...	further arguments passed to barplot.

Value

Invisibly returns the named importance vector (full length, before truncation to top_n).

See Also

[VI_SWORD](#), [plot.sword_flat](#)

Examples

```
set.seed(1)
X <- data.frame(matrix(rnorm(150 * 5), 150, 5,
  dimnames = list(NULL, paste0("x", 1:5))))
y <- X$x1 * 2 - X$x2 + rnorm(150)
forest <- SWORD(X, y, m = 10, oob = FALSE, verbose = FALSE)
plot_vi_sword(forest)
```

predict.sword_flat *Predict from a fitted SWORD forest*

Description

Predict from a fitted SWORD forest

Usage

```
## S3 method for class 'sword_flat'
predict(object, newdata, aggregate = TRUE,
        interval = c("none", "prediction"), level = 0.95, ...)
```

Arguments

object	sword_flat object from SWORD .
newdata	data.frame of predictors.
aggregate	Logical; if TRUE (default) return the ensemble mean. If FALSE return an $n \times m$ matrix of per-tree predictions.
interval	Character; "none" (default) or "prediction". When "prediction", returns a three-column matrix with columns fit, lwr, and upr from empirical quantiles across trees. Ignored when aggregate = FALSE.
level	Confidence level for the prediction interval (default 0.95). Ignored unless interval = "prediction".
...	ignored.

Value

A numeric vector (default), an $n \times m$ matrix (aggregate = FALSE), or a three-column matrix (interval = "prediction").

Examples

```
set.seed(1)
X <- data.frame(matrix(rnorm(150 * 5), 150, 5,
                      dimnames = list(NULL, paste0("x", 1:5))))
y <- X$x1 * 2 - X$x2 + rnorm(150)
forest <- SWORD(X, y, m = 10, oob = FALSE, verbose = FALSE)
preds <- predict(forest, X[1:5, ])
# Per-tree predictions matrix
mat <- predict(forest, X[1:5, ], aggregate = FALSE)
# 95% prediction intervals
pi95 <- predict(forest, X[1:5, ], interval = "prediction")
```

predict.tors_flat	<i>Predict from a fitted TORS tree</i>
-------------------	--

Description

Predict from a fitted TORS tree

Usage

```
## S3 method for class 'tors_flat'
predict(object, newdata, ...)
```

Arguments

object	tors_flat object from TORS .
newdata	data.frame of predictors.
...	ignored.

Value

Numeric vector of predictions.

Examples

```
set.seed(1)
X <- data.frame(matrix(rnorm(150 * 5), 150, 5,
                      dimnames = list(NULL, paste0("x", 1:5))))
y <- X$x1 * 2 - X$x2 + rnorm(150)
tree <- TORS(X, y, nmin = 10, cp = 0.01)
preds <- predict(tree, X[1:5, ])
```

SWORD	<i>Fit a SWORD forest</i>
-------	---------------------------

Description

Fits SWORD (Support vector machine Weighted Oblique Random Decision forests), a random forest of TORS oblique regression trees. Each tree is trained on a bootstrap sample; diversity is further increased by random predictor subsampling and optional randomisation of the number of features per split. Out-of-bag (oob) predictions and metrics are computed when oob = TRUE. Parallel fitting is supported via furrr when the caller sets a future plan before invoking SWORD.

Usage

```

SWORD(
  Covariates,
  y = NULL,
  data = NULL,
  nmin = 5,
  minleaf = 2,
  cp = 0,
  n_top_cor = 2,
  m = 100,
  rf_var = NULL,
  rand_ntopcor = TRUE,
  relation = "Pearson",
  weight_scheme = "scale",
  type_of_svm = "C-classification",
  cost_c = 1,
  cost_nu = 0.5,
  tolerance = 0.001,
  seed_bs = 25,
  parallel = FALSE,
  chunk = FALSE,
  n_chunks = NULL,
  n_workers = 1,
  oob = TRUE,
  verbose = FALSE,
  timeout_tree = 300
)

## S3 method for class 'sword_flat'
print(x, ...)

## S3 method for class 'sword_flat'
summary(object, ...)

```

Arguments

<code>Covariates</code>	data.frame of predictors, or a formula for the formula interface.
<code>y</code>	numeric response vector.
<code>data</code>	data.frame; required when <code>Covariates</code> is a formula.
<code>nmin</code>	minimum node size (default 5).
<code>minleaf</code>	minimum leaf size (default 2).
<code>cp</code>	complexity parameter (default 0). Unlike TORS , forests use <code>cp = 0</code> (full trees) by design: bootstrap sampling provides regularisation and unpruned trees maximise diversity.
<code>n_top_cor</code>	top-correlated features per split (default 2).
<code>m</code>	number of trees (default 100).

<code>rf_var</code>	features sub-sampled per split (default <code>ncol(Covariates)</code>).
<code>rand_ntopcor</code>	randomly draw top-feature cardinality (default <code>TRUE</code>).
<code>relation</code>	"Pearson" or "MI".
<code>weight_scheme</code>	"scale" or "robust".
<code>type_of_svm</code>	SVM type: "C-classification" uses <code>cost_c</code> ; "nu-classification" uses <code>cost_nu</code> (default "C-classification").
<code>cost_c</code>	SVM cost parameter C; used when <code>type_of_svm = "C-classification"</code> (default 1).
<code>cost_nu</code>	nu parameter; used when <code>type_of_svm = "nu-classification"</code> (default 0.5).
<code>tolerance</code>	convergence tolerance for the C++ solver (default 0.001).
<code>seed_bs</code>	integer seed offset for bootstrap samples (default 25).
<code>parallel</code>	if <code>TRUE</code> , fit trees in parallel via <code>furrr::future_map</code> (default <code>FALSE</code>). Set a future plan before calling.
<code>chunk</code>	if <code>TRUE</code> , process trees in chunks (default <code>FALSE</code>).
<code>n_chunks</code>	number of chunks; <code>NULL</code> = <code>ceiling(m / n_workers)</code> (default <code>NULL</code>).
<code>n_workers</code>	number of workers used only to derive the default <code>n_chunks</code> when <code>chunk = TRUE</code> and <code>n_chunks = NULL</code> (default 1). The actual parallelism is controlled by the future plan set by the caller.
<code>oob</code>	if <code>TRUE</code> , compute oob predictions and metrics (default <code>TRUE</code>).
<code>verbose</code>	if <code>TRUE</code> , print tree index during sequential fitting (default <code>FALSE</code>).
<code>timeout_tree</code>	max seconds per tree; <code>NULL</code> = no limit (default 300). Trees that exceed the limit are skipped and excluded from aggregation; the count of skipped trees is stored in <code>object\$n_skipped</code> and printed by <code>print.sword_flat</code> .
<code>x</code>	<code>sword_flat</code> object (for print).
<code>...</code>	ignored.
<code>object</code>	<code>sword_flat</code> object (for summary).

Value

An object of class "sword_flat".

References

- Carta, A. and Frigau, L. (in press). Oblique random forests for regression via weighted support vector machine. *Advances in Data Analysis and Classification*.
- Carta, A. and Frigau, L. (2025). Tree oblique for regression with weighted support vector machine. *Computational Statistics*, **40**, 5257–5291. doi:[10.1007/s0018002501647w](https://doi.org/10.1007/s0018002501647w)
- Yang, X., Song, Q. and Cao, A. (2005). Weighted support vector machine for data classification. In *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks (IJCNN)*. doi:[10.1109/IJCNN.2005.1555965](https://doi.org/10.1109/IJCNN.2005.1555965)
- Xu, T. et al. (2024). *WeightSVM: Subject/Instance Weighted Support Vector Machines*. R package version 1.7-16. doi:[10.32614/CRAN.package.WeightSVM](https://doi.org/10.32614/CRAN.package.WeightSVM)

See Also

[TORS](#), [predict.sword_flat](#), [VI_SWORD](#)

Examples

```
set.seed(1)
X <- data.frame(matrix(rnorm(150 * 5), 150, 5,
                      dimnames = list(NULL, paste0("x", 1:5))))
y <- X$x1 * 2 - X$x2 + rnorm(150)
forest <- SWORD(X, y, m = 10, oob = TRUE, verbose = FALSE)
print(forest)

# formula interface
forest2 <- SWORD(y ~ ., data = cbind(X, y = y), m = 10, verbose = FALSE)
```

TORS

Fit a TORS oblique regression tree

Description

Grows one Tree Oblique for Regression with weighted SVM (TORS). At each internal node a Weighted-SVM classifier with a linear kernel defines the oblique splitting hyperplane as a sparse linear combination of predictors. The tree is stored in a flat (parallel vector) representation for memory efficiency and fast prediction.

Usage

```
TORS(
  Covariates,
  y = NULL,
  data = NULL,
  nmin = 5,
  minleaf = round(nmin/3),
  cp = 0.01,
  n_perc = 1,
  n_top_cor = 2,
  rf_var = NULL,
  rand_ntopcor = FALSE,
  relation = "Pearson",
  weight_scheme = "scale",
  type_of_svm = "C-classification",
  cost_c = 1,
  cost_nu = 0.5,
  tolerance = 0.001
)
```

```
## S3 method for class 'tors_flat'
print(x, ...)
```

```
## S3 method for class 'tors_flat'
summary(object, ...)
```

Arguments

Covariates	data.frame of predictors (no response column), or a formula for the formula interface.
y	numeric response vector (length = nrow(Covariates)).
data	data.frame; required when Covariates is a formula.
nmin	minimum node size to attempt a split (default 5).
minleaf	minimum leaf size after a split (default round(nmin/3)).
cp	complexity parameter: minimum relative post-split deviance (default 0.01; use 0 to grow the full tree).
n_perc	number of candidate quantile thresholds per split (default 1).
n_top_cor	number of top-correlated features per split (default 2).
rf_var	features randomly sub-sampled per split (default ncol(Covariates)).
rand_ntopcor	if TRUE, randomly draw the cardinality of top features (default FALSE).
relation	feature selection criterion: "Pearson" or "MI".
weight_scheme	observation weighting: "scale" or "robust".
type_of_svm	SVM type: "C-classification" uses the cost parameter cost_c; "nu-classification" uses cost_nu (fraction of margin errors/support vectors, in (0, 1]).
cost_c	SVM cost parameter C; used when type_of_svm = "C-classification" (default 1).
cost_nu	nu parameter; used when type_of_svm = "nu-classification" (default 0.5).
tolerance	convergence tolerance for the C++ solver (default 0.001).
x	tors_flat object (for print).
...	ignored.
object	tors_flat object (for summary).

Value

An object of class "tors_flat".

References

- Carta, A. and Frigau, L. (2025). Tree oblique for regression with weighted support vector machine. *Computational Statistics*, **40**, 5257–5291. doi:[10.1007/s0018002501647w](https://doi.org/10.1007/s0018002501647w)
- Yang, X., Song, Q. and Cao, A. (2005). Weighted support vector machine for data classification. In *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks (IJCNN)*. doi:[10.1109/IJCNN.2005.1555965](https://doi.org/10.1109/IJCNN.2005.1555965)
- Xu, T. et al. (2024). *WeightSVM: Subject/Instance Weighted Support Vector Machines*. R package version 1.7-16. doi:[10.32614/CRAN.package.WeightSVM](https://doi.org/10.32614/CRAN.package.WeightSVM)

See Also

[SWORD](#), [predict.tors_flat](#)

Examples

```
set.seed(1)
X <- data.frame(matrix(rnorm(150 * 5), 150, 5,
                      dimnames = list(NULL, paste0("x", 1:5))))
y <- X$x1 * 2 - X$x2 + rnorm(150)
tree <- TORS(X, y, nmin = 10, cp = 0.01)
print(tree)

# formula interface
tree2 <- TORS(y ~ ., data = cbind(X, y = y), nmin = 10, cp = 0.01)
```

VI_SWORD

Oblique Impurity-Weighted Variable Importance (OIW-VI)

Description

Computes the Oblique Impurity-Weighted Variable Importance (OIW-VI) for a fitted SWORD forest. At each internal node the deviance reduction is distributed among the predictors involved in the oblique split proportionally to their normalised absolute coefficients in the standardised feature space. These node-level scores are then averaged across all trees.

Usage

```
VI_SWORD(forest, raw = FALSE)
```

Arguments

forest	sword_flat object from SWORD .
raw	logical; if TRUE return importances at the dummy-column level instead of the original variable level (default FALSE).

Details

When the forest was trained on data with factor or character columns, dummy-column contributions are automatically summed back to the original variable level (e.g. `cutFair + cutGood + ... -> cut`). Set `raw = TRUE` to get the raw dummy-column importances instead.

Value

Named numeric vector of importance scores, sorted descending.

Examples

```
set.seed(1)
X <- data.frame(matrix(rnorm(150 * 5), 150, 5,
                      dimnames = list(NULL, paste0("x", 1:5))))
y <- X$x1 * 2 - X$x2 + rnorm(150)
forest <- SWORD(X, y, m = 10, oob = FALSE, verbose = FALSE)
VI_SWORD(forest)
```

Index

pdp_sword, [2](#), [4](#)
plot.sword_flat, [3](#), [4](#), [6](#), [9](#)
plot.tors_flat, [5](#)
plot_oob_fit, [4](#), [6](#)
plot_tors_text, [5](#), [7](#), [8](#)
plot_tors_visnet, [5](#), [7](#), [8](#)
plot_vi_sword, [4](#), [9](#)
predict.sword_flat, [10](#), [14](#)
predict.tors_flat, [11](#), [16](#)
print.sword_flat, [13](#)
print.sword_flat (SWORD), [11](#)
print.tors_flat (TORS), [14](#)

summary.sword_flat (SWORD), [11](#)
summary.tors_flat (TORS), [14](#)
SWORD, [3](#), [4](#), [6](#), [9](#), [10](#), [11](#), [16](#)

TORS, [5](#), [7](#), [8](#), [11](#), [12](#), [14](#), [14](#)

VI_SWORD, [9](#), [14](#), [16](#)