Package: svmodt (via r-universe)

July 1, 2026

Type Package

Title Linear SVM-Based Recursive Decision Trees

Version 0.1.0

Description Implements Support Vector Machine Oblique Decision Trees (SVMODT). Recursively builds classification trees using linear Support Vector Machines (SVM) hyperplanes at each node instead of axis-parallel splits, creating oblique decision boundaries. Features include multiple feature selection methods, dynamic feature subset strategies, class weight support for imbalanced datasets, pruning and feature penalization.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Suggests knitr, rmarkdown, bookdown, testthat (>= 3.0.0), rpart, rsample, gridExtra, tidyr, kableExtra, palmerpenguins, dplyr

VignetteBuilder knitr

Depends R (>= 3.5)

Imports rlang, e1071, FSelectorRcpp, ggplot2

RoxygenNote 7.3.3

Config/testthat/edition 3

URL https://github.com/AneeshAgarwala/svmodt

BugReports https://github.com/AneeshAgarwala/svmodt/issues

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

Date/Publication 2026-06-24 09:09:16 UTC

RemoteUrl https://github.com/aneeshagarwala/svmodt

RemoteRef HEAD

RemoteSha 1ec22a7fc9d15f9e24aca6126ff8fa7f4e6ea9b4

Contents

`plot.svmodt_node` *Plot method for svmodt_node objects*

Description

Thin S3 wrapper that dispatches to `plot_boundary` or `plot_surface` depending on `plot.type`.

Usage

```
## S3 method for class 'svmodt_node'
plot(
  x,
  y = NULL,
  ...,
  data = NULL,
  response = NULL,
  plot.type = c("surface", "boundary"),
  features = NULL,
  max_depth = NULL,
  check_accuracy = TRUE,
  resolution = NULL
)
```

Arguments

`x`	An `svmodt_node` returned by `svm_split`.
`y`	Ignored; present only to satisfy the `graphics::plot` generic signature.
`...`	Currently unused.
`data`	The original training data frame (required).
`response`	Character string naming the response column (required).
`plot.type`	One of `"surface"` (default) or `"boundary"`.
`features`	Length-2 character vector of axis features (`"surface"` only; default uses root node features).
`max_depth`	Maximum depth to visualize (`"boundary"` only; default `NULL` = full tree).
`check_accuracy`	Logical; show per-node accuracy (`"boundary"` only; default `TRUE`).
`resolution`	Grid resolution per axis. Default `100` for `"boundary"`, `200` for `"surface"`.

Value

- "boundary": invisibly returns the list from plot_boundary.
- "surface": invisibly returns the **ggplot2** object from plot_surface.

Examples

```
tree <- svm_split(wdbc, response = "diagnosis", max_depth = 3)

# All-node boundary panels - prints first, returns list
viz <- plot(tree,
  data = wdbc, response = "diagnosis",
  plot.type = "boundary"
)
viz$plots[[2]] # second node

# Global decision surface
plot(tree,
  data = wdbc, response = "diagnosis",
  plot.type = "surface"
)

# Surface with explicit feature axes
plot(tree,
  data = wdbc, response = "diagnosis",
  plot.type = "surface",
  features = c("radius_mean", "concavity_mean")
)
```

---

predict.svmodt_node *Predict method for svmodt_node objects*

Description

Predict method for svmodt_node objects

Usage

```
## S3 method for class 'svmodt_node'
predict(object, newdata, return_probs = FALSE, calibrate_probs = TRUE, ...)
```

Arguments

object	An object of class svmodt_node.
newdata	A data frame of new predictor values.
return_probs	Logical; if TRUE, returns predictions and probabilities.

`calibrate_probs`
: Logical; if `TRUE`, uses logistic calibration on decision values.

`...`
: Currently unused.

Value

If `return_probs = FALSE` (the default), a character vector of predicted class labels, one element per row of `newdata`.

If `return_probs = TRUE`, a named list with two elements:

predictions Character vector of predicted class labels (length = `nrow(newdata)`).

probabilities Numeric matrix of class probabilities with `nrow(newdata)` rows and one column per class. Column names are the class labels; each row sums to 1. When `calibrate_probs = TRUE`, probabilities are derived from the SVM decision value via logistic calibration; otherwise empirical class frequencies at the leaf node are used.

Examples

```
# Train DTSVM tree
tree <- svm_split(
  data = wdbc,
  response = "diagnosis",
  max_depth = 3,
  max_features = 2,
  feature_method = "cor"
)

# Predict on WDBC data (returns a character vector of class labels)
preds <- predict(tree, newdata = wdbc)

# Predict with probabilities and logistic calibration
result <- predict(tree, newdata = wdbc,
  return_probs = TRUE, calibrate_probs = TRUE
)
head(result$predictions)
head(result$probabilities)
```

`print.svmodt_node` *' Print method for svmodt_node objects*

Description

' Print method for svmodt_node objects

Usage

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

Arguments

x	An object of class svmodt_node.
...	Further arguments passed to print_svm_tree.

Value

Invisibly returns x (the svmodt_node object), called for its side effect of printing a human-readable summary of the tree structure to the console.

Examples

```
tree <- svm_split(
  data = wdbc,
  response = "diagnosis",
  max_features = 2,
  max_depth = 3,
  min_samples = 5,
  feature_method = "random",
  verbose = TRUE
)
print(tree)
```

---

svm_split *Build an Oblique Decision Tree Using SVM Splits*

Description

Constructs a decision tree where each internal node uses a Support Vector Machine (SVM) to determine the split. Supports dynamic feature selection, feature penalization, scaling, and class weighting.

Usage

```
svm_split(
  data,
  response,
  depth = 1,
  max_depth = 10,
  min_samples = 5,
  max_features = NULL,
  feature_method = c("random", "mutual", "cor"),
  impurity_measure = c("entropy", "gini"),
  max_features_strategy = c("constant", "random", "decrease"),
  max_features_decrease_rate = 0.8,
  max_features_random_range = c(0.3, 1),
  penalize_used_features = FALSE,
```

```
  feature_penalty_weight = 0.5,
  n_subsets = 1,
  used_features = character(0),
  class_weights = c("none", "balanced", "custom"),
  custom_class_weights = NULL,
  min_impurity_decrease = 0.001,
  verbose = FALSE,
  all_classes = NULL,
  ...
)
```

Arguments

`data`	A data frame containing predictors and the response variable.
`response`	Character string specifying the response column in ‘data‘. All other columns are treated as predictors.
`depth`	Integer indicating the current recursion depth (used internally; default is 1).
`max_depth`	Maximum depth of the tree.
`min_samples`	Minimum number of samples required to attempt a split.
`max_features`	Maximum number of features to consider at each split.
`feature_method`	Feature selection method at each node. One of: • ‘"random"‘: randomly select features, • ‘"mutual"‘: select based on mutual information with the response, • ‘"cor"‘: select based on correlation with the response.
`impurity_measure`	Information Gain evaluation criteria • ‘"gini"‘: use Gini ratio • ‘"entropy"‘: use Shannon entropy
`max_features_strategy`	Strategy to adjust the number of features per node: • ‘"constant"‘: keep ‘max_features‘ constant, • ‘"decrease"‘: reduce features with depth, • ‘"random"‘: randomly vary number of features within a range.
`max_features_decrease_rate`	Numeric fraction for decreasing features if ‘max_features_strategy = "decrease"‘.
`max_features_random_range`	Numeric vector of length 2 specifying min and max fraction of features if ‘max_features_strategy = "random"‘.
`penalize_used_features`	Logical; if TRUE, features used in ancestor nodes are penalized to encourage diversity.
`feature_penalty_weight`	Numeric (0<U+2013>1) weight for penalizing previously used features.

`n_subsets` Number of Evaluated Random Feature combinations at each node when ‘feature_method = "random‘

`used_features` Character vector of features already used in ancestor nodes (used internally).

`class_weights` Character string specifying how to handle class imbalance. One of:

- ‘"none"‘: no weighting,
- ‘"balanced"‘: weight classes inversely proportional to their frequency,
- ‘"custom"‘: use ‘custom_class_weights‘.

`custom_class_weights` Optional named numeric vector specifying custom weights per class.

`min_impurity_decrease` Required decrease in impurity by a split to be considered valid

`verbose` Logical; if TRUE, prints information about each node during tree construction.

`all_classes` Optional character vector of all possible response classes (used internally).

`...` Additional arguments passed to the underlying SVM fitting function.

Details

This function recursively splits the dataset using an SVM at each node. Splitting stops when maximum depth is reached, the node contains fewer than ‘min_samples‘, or all samples belong to the same class. Features are scaled and selected dynamically at each node, and previously used features can be penalized to promote diversity. Class weighting schemes support handling imbalanced datasets. This approach allows construction of an **oblique decision tree**, where splits are linear hyperplanes rather than axis-aligned.

Value

A nested list representing the decision tree. Each node contains:

is_leaf Logical; TRUE if the node is a leaf.

model Fitted SVM model at this node (for internal nodes).

features Vector of features selected for this node.

scaler Scaling information used at this node.

left Left child node (decision value > 0).

right Right child node (decision value <U+2264> 0).

depth Depth of this node in the tree.

n Number of samples at this node.

max_features_used Number of features considered at this node.

penalty_applied Logical; TRUE if feature penalization was applied.

class_weights_used Class weights applied at this node.

Examples

```
data(wdbc)
tree <- svm_split(
  data = wdbc,
  response = "diagnosis",
  max_depth = 3,
  min_samples = 5,
  feature_method = "random",
  verbose = TRUE
)
```

`trace_path` *Trace the prediction path of a sample through an svmodt tree*

Description

Generic function that walks the tree for a single row of new data, printing the SVM decision value and chosen branch at every internal node and the final predicted class at the leaf.

Usage

```
trace_path(object, ...)

## S3 method for class 'svmodt_node'
trace_path(object, sample_data, sample_idx = 1, ...)
```

Arguments

`object`	An `svmodt_node` returned by `svm_split`.
`...`	Currently unused.
`sample_data`	A data frame of new predictor values (one or more rows).
`sample_idx`	Integer; which row to trace (default `1`).

Value

Invisibly returns the predicted class label (character string).

Methods (by class)

- `trace_path(svmodt_node)`: Method for `svmodt_node` objects.

Examples

```
tree <- svm_split(wdbc, response = "diagnosis", max_depth = 3)
trace_path(tree, wdbc, sample_idx = 5)
```

wdbc *Wisconsin Diagnostic Breast Cancer Dataset*

Description

The WDBC dataset contains quantitative measurements from digitized images of fine needle aspirates (FNA) of breast masses. It is commonly used for classification tasks to distinguish between benign and malignant tumors.

Usage

```
wdbc
```

Format

A data frame with 569 rows and 32 columns:

radius_mean Mean of radius

radius_se Standard error of radius

radius_worst Worst (largest) radius

texture_mean Mean of texture

texture_se Standard error of texture

texture_worst Worst texture

perimeter_mean Mean of perimeter

perimeter_se Standard error of perimeter

perimeter_worst Worst perimeter

area_mean Mean area

area_se Standard error of area

area_worst Worst area

smoothness_mean Mean smoothness

smoothness_se Standard error of smoothness

smoothness_worst Worst smoothness

compactness_mean Mean compactness

compactness_se Standard error of compactness

compactness_worst Worst compactness

concavity_mean Mean concavity

concavity_se Standard error of concavity

concavity_worst Worst concavity

concave.points_mean Mean concave points

concave.points_se Standard error of concave points

concave.points_worst Worst concave points
symmetry_mean Mean symmetry
symmetry_se Standard error of symmetry
symmetry_worst Worst symmetry
fractal_dimension_mean Mean fractal dimension
fractal_dimension_se Standard error of fractal dimension
fractal_dimension_worst Worst fractal dimension
diagnosis Factor with levels 'B' and 'M'

Source

Dr. William H. Wolberg, W. Nick Street, and Olvi L. Mangasarian, University of Wisconsin<U+2013>Madison. Original dataset available at: <https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic>

---

wine *Wine Dataset*

Description

The Wine dataset contains the results of a chemical analysis of wines derived from three different cultivars grown in the same region of Italy. The dataset is commonly used for multiclass classification tasks, where the objective is to identify the cultivar of origin based on physicochemical properties.

Usage

```
wine
```

Format

A data frame with 178 rows and 14 columns:

class Factor with levels 1, 2, and 3 indicating cultivar
alcohol Alcohol content
malic_acid Malic acid concentration
ash Ash content
alcalinity_of_ash Alcalinity of ash
magnesium Magnesium content
total_phenols Total phenols
flavanoids Flavonoid content
nonflavanoid_phenols Nonflavanoid phenols
proanthocyanins Proanthocyanin content
color_intensity Color intensity
hue Hue
od280_od315 OD280/OD315 of diluted wines
proline Proline concentration

Source

Aeberhard, S. & Forina, M. (1992). Wine Dataset. UCI Machine Learning Repository. Original dataset available at: <https://archive.ics.uci.edu/dataset/109/wine>

Index