Package: ppmSDR (via r-universe)

October 26, 2026

Type Package

Title Penalized Principal Machine for Sufficient Dimension Reduction

Version 3.0.1

Author Jungmin Shin [aut, cre] (Department of Biomedical Informatics, The Ohio State University), Seung Jun Shin [aut] (Department of Statistics, Korea University)

Maintainer Jungmin Shin <c16267@gmail.com>

Description A unified, computation-friendly framework for penalized principal machines (P2M), a class of sparse sufficient dimension reduction (SDR) estimators for regression and binary classification. Principal machines (PM) estimate the central subspace by solving a family of convex-loss problems over several cutoffs; their penalized counterparts (P2M) add a row-group sparsity penalty so that dimension reduction and variable selection are performed simultaneously. All estimators are fitted by a single group coordinate descent (GCD) algorithm that accommodates least squares, logistic, asymmetric least squares, L2-hinge, hinge (support vector machine, SVM) and quantile losses, together with the least absolute shrinkage and selection operator (LASSO), the smoothly clipped absolute deviation (SCAD) penalty and the minimax concave penalty (MCP). Methods are described in Li, Artemiou and Li (2011) <doi:10.1214/11-AOS932>, Shin and Artemiou (2017) <doi:10.1016/j.csda.2016.12.003>, Artemiou, Dong and Shin (2021) <doi:10.1016/j.patcog.2020.107768> and Breheny and Huang (2015) <doi:10.1007/s11222-013-9424-2>.

Depends R (>= 4.1.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports stats, Matrix, grpreg, energy

Suggests MASS, knitr, rmarkdown, testthat (>= 3.0.0)

URL https://github.com/c16267/ppmSDR

BugReports https://github.com/c16267/ppmSDR/issues

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libgsl0-dev

Repository https://c16267.r-universe.dev

Date/Publication 2026-09-26 21:06:29 UTC

RemoteUrl https://github.com/c16267/ppmsdr

RemoteRef HEAD

RemoteSha c76079ac86dba07b8314333145ff0d33648ae7b8

Contents

boston . . . 2
ppm . . . 4
ppm_tune . . . 8
print.ppm . . . 11
print.ppm_tune . . . 12
summary.ppm . . . 12
wdbc . . . 13

Index **15**

`boston` *Boston Housing Data*

Description

The Boston housing data of Harrison and Rubinfeld (1978), as distributed in **MASS**. Following the real-data analysis in the package's accompanying article, the binary Charles-River dummy variable (`chas`) is removed, leaving twelve continuous predictors and the response `medv`. Values are on their natural scale; standardize them before fitting (see Examples).

Usage

```
boston
```

Format

A data frame with 506 rows and 13 variables:

crim per-capita crime rate by town.

zn proportion of residential land zoned for lots over 25,000 sq ft.

indus proportion of non-retail business acres per town.

nox nitrogen-oxides concentration (parts per 10 million).

rm average number of rooms per dwelling.

age proportion of owner-occupied units built prior to 1940.

dis weighted mean of distances to five Boston employment centres.

rad index of accessibility to radial highways.

tax full-value property-tax rate per $10,000.

ptratio pupil-teacher ratio by town.

b $1000(Bk - 0.63)^2$, where Bk is the proportion of the Black population by town. See the note below.

lstat lower-status percentage of the population.

medv median value of owner-occupied homes in $1000s (the response).

Note

The variable `b` embeds a racist premise from the original 1978 study and is retained only to reproduce the published analysis. It should not be used uncritically; see the discussion in the documentation of `MASS::Boston`.

Source

Harrison, D. and Rubinfeld, D. L. (1978) Hedonic prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5, 81–102. Distributed in the **MASS** package.

Examples

```
data(boston)
x <- scale(as.matrix(boston[, setdiff(names(boston), "medv")]))
y <- as.numeric(scale(boston$medv))
fit <- ppm(x, y, loss = "lssvm", penalty = "grSCAD", lambda = 8e-3)
summary(fit, d = 2)
```

ppm *Penalized Principal Machine for Sufficient Dimension Reduction*

Description

Fits a penalized principal machine (P2M), a sparse sufficient dimension reduction (SDR) estimator, through a single group coordinate descent (GCD) engine. A principal machine (PM) estimates the basis of the central subspace by solving a family of convex-loss problems over several cutoffs (slices); the penalized version adds a row-group sparsity penalty so that dimension reduction and variable selection are performed simultaneously.

Usage

```
ppm(
  x,
  y,
  loss = "lssvm",
  H = 10,
  C = 1,
  lambda = 0.01,
  gamma = 3.7,
  penalty = c("grSCAD", "grLasso", "grMCP"),
  max.iter = 100,
  line.search = NULL,
  max.halving = 10,
  ...
)
```

Arguments

x	A numeric matrix or data frame of predictors, of dimension n (observations) by p (variables).
y	A response vector of length n. For continuous-type losses a numeric vector is expected; for weighted losses (those whose name starts with "w") a two-class response is expected and is recoded internally to $\{-1, +1\}$.
loss	Character string selecting the loss function. One of "lssvm", "wlssvm", "svm", "wsvm", "l2svm", "wl2svm", "logit", "wlogit", "asls", "qr". Default "lssvm".
H	Number of cutoffs (slices); a single integer >= 2. Default 10.
C	Positive cost parameter that balances the loss against the covariance term. Default 1.
lambda	Positive regularization parameter controlling sparsity. Default 0.01. In practice lambda should be selected by cross-validation.
gamma	Concavity parameter of the SCAD/MCP penalty; must exceed 2. Default 3.7.
penalty	Penalty type: "grSCAD" (default), "grLasso" or "grMCP".
max.iter	Maximum number of outer (iterative GCD / MM-GCD) iterations. Default 100.

`line.search` Logical or `NULL`. Whether to apply the step-halving line search on the original penalized objective described in the *Algorithms* section. The default `NULL` selects `TRUE` for the iterative GCD losses `"logit"`, `"wlogit"`, `"l2svm"` and `"wl2svm"`, and `FALSE` for all other losses (`"asls"`, the MM-GCD losses `"svm"`, `"wsvm"`, `"qr"`, and the exact GCD losses). Setting `FALSE` for an iterative GCD loss gives the plain (undamped) updates used in the numerical studies of the accompanying article; setting `TRUE` for `"asls"` or for an MM-GCD loss adds the safeguard on top of the corresponding quadratic step. The argument is ignored, with a warning, for the exact GCD losses `"lssvm"` and `"wlssvm"`.

`max.halving` Maximum number of step halvings $m_{\max}$ tried at each iteration when `line.search = TRUE`. Default `10`.

`...` Additional arguments passed to the underlying solver. The most useful are `ridge`, a small non-negative ridge constant added for numerical stability, which is accepted by the iterative solvers (`logit`, `wlogit`, `svm`, `wsvm`, `qr`, `lssvm`, `wlssvm`, `wl2svm`), and `eps`, the positive floor of the quadratic majorizer for the MM-GCD losses (`svm`, `wsvm`: default `1e-6`; `qr`: default `0.05 * sd(y)`, see the *Algorithms* section).

Details

`ppm()` is a single front-end that dispatches to the loss-specific solver selected by `loss`. Two families are supported, following the taxonomy of Shin and Shin (2024):

- **Response-based PM (RPM)** for a continuous response, where the loss is fixed and the pseudo-response varies across slices: `"lssvm"` (least squares, P2LSM), `"l2svm"` (L2-hinge, P2L2M), `"svm"` (hinge, P2SVM), `"logit"` (logistic, P2LR), `"asls"` (asymmetric least squares, P2AR) and `"qr"` (quantile, P2QR).
- **Loss-based PM (LPM)** for a binary response coded internally as $\{-1, +1\}$, where the loss varies across slices: `"wlssvm"` (P2WLSM), `"wl2svm"` (P2WL2M), `"wsvm"` (P2WSVM) and `"wlogit"` (P2WLR).

Acronyms used above: SDR (sufficient dimension reduction), PM (principal machine), P2M (penalized principal machine), GCD (group coordinate descent), SVM (support vector machine). The penalty `penalty` is one of the group LASSO (least absolute shrinkage and selection operator), the group SCAD (smoothly clipped absolute deviation) or the group MCP (minimax concave penalty), passed as `"grLasso"`, `"grSCAD"` or `"grMCP"`.

The basis of the central subspace is estimated by the leading eigenvectors of the working matrix $M = \sum_{k=1}^{H} \beta_k \beta_k^\top$, where β_k is the slope estimated at the k-th cutoff.

Value

An object of S3 class `"ppm"`, a list containing:

M the estimated working matrix (a `p` by `p` symmetric matrix).

evalues, evectors the eigenvalues and eigenvectors of `M`; the leading `d` eigenvectors estimate the basis of the central subspace.

theta the estimated coefficient matrix Θ of dimension `(p + 1)` by `H - 1`, whose `k`-th column holds the intercept (first row) and the slope β_k of the `k`-th machine.

x, y the (validated) input data.

loss, penalty, lambda, gamma, C, H, max.iter the fitting settings.

algorithm `"GCD"`, `"iterative GCD"` or `"MM-GCD"`.

line.search, max.halving the line-search settings actually used.

iter the number of outer iterations performed.

status `"converged"`, `"max.iter"` or `"line.search.halted"`.

objective when `line.search = TRUE`, the value of the penalized objective $Q(\theta^{(t)})$ after each accepted iteration (non-increasing by construction); otherwise `NULL`.

n.halving when `line.search = TRUE`, the number of halvings m_t at each iteration (`NA` at a halted iteration); otherwise `NULL`.

ytype `"continuous"` or `"binary"`.

n, p the sample size and the number of predictors.

call the matched call.

Algorithms and the step-halving line search

The solvers fall into three classes, and the class determines the default of `line.search`:

- **GCD** (`"lssvm"`, `"wlssvm"`): the squared loss gives an exact group penalized least squares problem, solved by a single GCD run. There is no outer iteration and `line.search` does not apply (it is ignored with a warning if set to `TRUE`).
- **Iterative GCD** (`"logit"`, `"wlogit"`, `"asls"`, `"l2svm"`, `"wl2svm"`): the loss is replaced at every outer iteration by a local quadratic approximation (IRLS for the logistic loss; residual signs or the active set frozen for the asymmetric squared and squared hinge losses). Such an approximation is not a majorizer, so a plain GCD update does not by itself guarantee descent of the penalized objective $Q(\theta)$. For `"logit"`, `"wlogit"`, `"l2svm"` and `"wl2svm"` the default `line.search = TRUE` therefore pairs every update with a step-halving safeguard. For `"asls"` (P2AR) the default is `line.search = FALSE`, i.e. the plain updates used in the numerical studies of the accompanying article; the safeguard can be switched on, and it is advisable to do so when a large cost parameter is used (the plain sweep is stable for moderate values, roughly `C <= 5`; for larger `C` it can diverge and the solver then stops with an informative error).
- **MM-GCD** (`"svm"`, `"wsvm"`, `"qr"`): the hinge and check losses are replaced by quadratic majorizers, so descent follows from the majorization-minimization argument and the default is `line.search = FALSE`. The safeguard can still be switched on. The majorizers use a floor `eps` that safeguards a zero margin (hinge) or a zero residual (check loss). For `"svm"` and `"wsvm"` the margin is scale-free and `eps = 1e-6`. For `"qr"` the residual has the scale of the response and, since the quantile-regression solution interpolates `p + 1` observations exactly, a tiny absolute floor makes the MM weights `1 / (4 eps)` explode as the iterations approach the solution; the default is therefore relative, `eps = 0.05 * sd(y)`, which keeps the iterations stable and can be overridden through `...`.

With `line.search = TRUE`, if $\hat{\theta}^{(t)}$ denotes the candidate returned by the GCD step at iteration t, the accepted update is

$$\theta^{(t+1)} = \theta^{(t)} + 2^{-m_t}\{\hat{\theta}^{(t)} - \theta^{(t)}\},$$

where $m_t \geq 0$ is the smallest integer for which $Q(\theta^{(t+1)}) \leq Q(\theta^{(t)})$, and Q is the original penalized P2M objective

$$Q(\theta) = \sum_{k=1}^{h} \left[\theta_k^\top \tilde{\Sigma} \theta_k + \frac{c}{n} \sum_{i=1}^{n} w_{ik} L_k(\tilde{y}_{ik}, \theta_k^\top \tilde{x}_i) \right] + \sum_{j=1}^{p} p_\lambda(\|\theta_{(j)}\|),$$

evaluated on the objective itself and not on the quadratic surrogate. If no acceptable step is found within `max.halving` halvings, the current iterate is retained and the algorithm stops. The returned `status` is then `"converged"` when the rejected candidate step already satisfied the stopping rule below (the plain iteration would have stopped there as well), and `"line.search.halted"` otherwise, meaning that the GCD direction failed to decrease Q at a non-converged iterate and the last accepted iterate is returned. No warning is issued; inspect `status`, `iter` and `n.halving` (also shown by `print.ppm()`) to see whether and where the safeguard intervened. The safeguard guarantees monotone non-increase of Q along the accepted iterates; it coincides with the plain update ($m_t = 0$) at every iteration at which the plain update does not increase Q, so `line.search = FALSE` reproduces the plain iterative GCD exactly.

All iterative solvers start at $\theta^{(0)} = 0$ and stop once $\|\theta^{(t+1)} - \theta^{(t)}\|_\infty / (1 + \|\theta^{(t)}\|_\infty) < 10^{-5}$ or after `max.iter` outer iterations.

References

Li, B., Artemiou, A. and Li, L. (2011) Principal support vector machines for linear and nonlinear sufficient dimension reduction. *The Annals of Statistics*, 39(6), 3182–3210. doi:10.1214/11AOS932

Shin, S. J. and Artemiou, A. (2017) Penalized principal logistic regression for sparse sufficient dimension reduction. *Computational Statistics & Data Analysis*, 111, 48–58. doi:10.1016/j.csda.2016.12.003

Breheny, P. and Huang, J. (2015) Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors. *Statistics and Computing*, 25, 173–187. doi:10.1007/s1122201394242

See Also

`print.ppm()`, `summary.ppm()`

Examples

```
set.seed(1)
n <- 1000; p <- 10
B <- matrix(0, p, 2); B[1, 1] <- B[2, 2] <- 1
x <- matrix(rnorm(n * p), n, p)
y <- (x %*% B[, 1]) / (0.5 + (x %*% B[, 2] + 1)^2) + 0.2 * rnorm(n)

## penalized principal least-squares SVM (P2LSM) with the group SCAD penalty
fit <- ppm(x, y, loss = "lssvm", penalty = "grSCAD", lambda = 0.01)
round(fit$evectors[, 1:2], 3)
print(fit)
summary(fit)

## penalized principal logistic regression (P2LR): iterative GCD, so the
## step-halving line search is on by default
```

```
fit_lr <- ppm(x, y, loss = "logit", penalty = "grSCAD", lambda = 0.01)
fit_lr$status
fit_lr$n.halving            # m_t at each iteration (0 = plain update)
all(diff(fit_lr$objective) <= 0)   # Q is non-increasing

## the plain (undamped) updates used in the article's numerical studies
fit_lr0 <- ppm(x, y, loss = "logit", penalty = "grSCAD", lambda = 0.01,
               line.search = FALSE)


## binary response with a two-dimensional central subspace spanned by
## (x1, x2): penalized principal asymmetric least squares (P2AR), plain
## updates by default, and penalized principal weighted logistic regression
yb <- sign(x[, 1] + x[, 2]^3 / 3 + 0.2 * rnorm(n))
fitw <- ppm(x, yb, loss = "asls", penalty = "grSCAD", lambda = 0.2)
round(fitw$evectors[, 1:2], 3)
print(fitw)
summary(fitw)
fit_wlr <- ppm(x, yb, loss = "wlogit", penalty = "grSCAD", lambda = 0.005)
summary(fit_wlr)


data(boston)
xb <- scale(as.matrix(boston[, setdiff(names(boston), "medv")]))
yb <- as.numeric(scale(boston$medv))
fit_b <- ppm(xb, yb, loss = "lssvm", penalty = "grSCAD", lambda = 8e-3)
summary(fit_b, d = 2)
```

---

ppm_tune *Cross-Validation for the Penalized Principal Machine Tuning Parameter*

Description

Selects the sparsity parameter `lambda` of a penalized principal machine by K-fold cross-validation. For each candidate `lambda`, the model is fitted on the training folds and the held-out distance correlation (dCor; Szekely, Rizzo and Bakirov, 2007) between the response and the predictors projected onto the estimated `d`-dimensional central subspace is recorded. The `lambda` maximizing the average held-out dCor is returned, following the criterion of Shin, Wu, Zhang and Liu (2017).

Usage

```
ppm_tune(
  x,
  y,
  loss = "lssvm",
  d = 2,
```

```
  H = 10,
  C = 1,
  gamma = 3.7,
  penalty = c("grSCAD", "grLasso", "grMCP"),
  max.iter = 100,
  line.search = NULL,
  max.halving = 10,
  n.fold = 5,
  lambda = NULL,
  nlambda = 20,
  lambda.max = 0.1,
  lambda.min.ratio = 0.001,
  verbose = FALSE,
  ...
)
```

Arguments

x	A numeric matrix or data frame of predictors, of dimension n (observations) by p (variables).
y	A response vector of length n. For continuous-type losses a numeric vector is expected; for weighted losses (those whose name starts with "w") a two-class response is expected and is recoded internally to $\{-1, +1\}$.
loss	Character string selecting the loss function. One of "lssvm", "wlssvm", "svm", "wsvm", "l2svm", "wl2svm", "logit", "wlogit", "asls", "qr". Default "lssvm".
d	Working structural dimension used to form the held-out sufficient predictors. Default 2.
H	Number of cutoffs (slices); a single integer >= 2. Default 10.
C	Positive cost parameter that balances the loss against the covariance term. Default 1.
gamma	Concavity parameter of the SCAD/MCP penalty; must exceed 2. Default 3.7.
penalty	Penalty type: "grSCAD" (default), "grLasso" or "grMCP".
max.iter	Maximum number of outer (iterative GCD / MM-GCD) iterations. Default 100.
line.search	Logical or NULL. Whether to apply the step-halving line search on the original penalized objective described in the *Algorithms* section. The default NULL selects TRUE for the iterative GCD losses "logit", "wlogit", "l2svm" and "wl2svm", and FALSE for all other losses ("asls", the MM-GCD losses "svm", "wsvm", "qr", and the exact GCD losses). Setting FALSE for an iterative GCD loss gives the plain (undamped) updates used in the numerical studies of the accompanying article; setting TRUE for "asls" or for an MM-GCD loss adds the safeguard on top of the corresponding quadratic step. The argument is ignored, with a warning, for the exact GCD losses "lssvm" and "wlssvm".
max.halving	Maximum number of step halvings $m_{\max}$ tried at each iteration when line.search = TRUE. Default 10.
n.fold	Number of cross-validation folds. Default 5. For reproducible folds, call set.seed() before ppm_tune().

`lambda` Optional numeric vector of candidate values. If `NULL` (default), a log-spaced grid of length `nlambda` is built from `lambda.max` down to `lambda.max * lambda.min.ratio`.

`nlambda` Length of the automatically generated grid. Default `20`.

`lambda.max` Largest candidate value in the generated grid. Default `0.1`.

`lambda.min.ratio` Ratio of the smallest to the largest candidate in the generated grid. Default `1e-3`.

`verbose` Logical; if `TRUE`, progress is reported via `message()`. Default `FALSE`.

`...` Additional arguments passed to the underlying solver. The most useful are `ridge`, a small non-negative ridge constant added for numerical stability, which is accepted by the iterative solvers (`logit`, `wlogit`, `svm`, `wsvm`, `qr`, `lssvm`, `wlssvm`, `wl2svm`), and `eps`, the positive floor of the quadratic majorizer for the MM-GCD losses (`svm`, `wsvm`: default `1e-6`; `qr`: default `0.05 * sd(y)`, see the *Algorithms* section).

Details

The line-search settings `line.search` and `max.halving` are resolved once, exactly as in `ppm()` (the default `NULL` turns the step-halving safeguard on for `"logit"`, `"wlogit"`, `"l2svm"` and `"wl2svm"` only), and are then used both for every cross-validation fit and for the final refit on the full data.

Value

An object of S3 class `"ppm_tune"`, a list with elements

opt.lambda the selected value of `lambda`.

lambda the candidate grid (decreasing).

dcor the average held-out distance correlation at each candidate.

fit a `ppm()` object refitted on all data at `opt.lambda`.

d, loss, penalty, n.fold, line.search, max.halving, call the settings and the matched call.

References

Shin, S. J., Wu, Y., Zhang, H. H. and Liu, Y. (2017) Principal weighted support vector machines for sufficient dimension reduction in binary classification. *Biometrika*, 104(1), 67–81. doi:10.1093/biomet/asw057

Szekely, G. J., Rizzo, M. L. and Bakirov, N. K. (2007) Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, 35(6), 2769–2794. doi:10.1214/009053607000000505

See Also

`ppm()`

Examples

```
set.seed(1)
n <- 1000; p <- 10
x <- matrix(rnorm(n * p), n, p)
y <- x[, 1] / (0.5 + (x[, 2] + 1)^2) + 0.2 * rnorm(n)
cv <- ppm_tune(x, y, loss = "lssvm", d = 2, n.fold = 5,
               nlambda = 10, lambda.max = 0.02)
cv$opt.lambda
summary(cv$fit, d = 2)

## iterative GCD loss: the step-halving safeguard is used in every CV fit
cv_lr <- ppm_tune(x, y, loss = "logit", d = 2, n.fold = 5, nlambda = 5)
cv_lr$fit$status
```

`print.ppm` *Print a fitted penalized principal machine*

Description

Print a fitted penalized principal machine

Usage

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

Arguments

`x`	An object of class `"ppm"` returned by `ppm()`.
`...`	Currently ignored.

Value

The input object `x`, invisibly.

See Also

`ppm()`, `summary.ppm()`

Examples

```
set.seed(1)
x <- matrix(rnorm(200 * 6), 200, 6)
y <- x[, 1] / (0.5 + (x[, 2] + 1)^2) + 0.2 * rnorm(200)
fit <- ppm(x, y, loss = "lssvm", lambda = 0.01)
print(fit)
```

print.ppm_tune *Print a penalized principal machine cross-validation result*

Description

Print a penalized principal machine cross-validation result

Usage

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

Arguments

x	An object of class "ppm_tune" returned by ppm_tune().
...	Currently ignored.

Value

The input object x, invisibly.

See Also

ppm_tune()

summary.ppm *Summarize a fitted penalized principal machine*

Description

Reports the estimated basis of the central subspace for a given working dimension d, together with the predictors selected by the row-group penalty (those whose loadings are non-zero across the leading d directions).

Usage

```
## S3 method for class 'ppm'
summary(object, d = 2, tol = 1e-06, ...)
```

Arguments

object	An object of class "ppm" returned by ppm().
d	Working structural dimension; the number of leading eigenvectors to report. Default 2.
tol	Tolerance below which a row L2-norm is treated as zero when determining the selected variables. Default 1e-6.
...	Currently ignored.

Value

Invisibly, a list with elements `d`, `basis` (the `p` by `d` estimated basis), `selected` (indices of the selected predictors) and `n.selected`.

See Also

`ppm()`, `print.ppm()`

Examples

```
set.seed(1)
x <- matrix(rnorm(200 * 6), 200, 6)
y <- x[, 1] / (0.5 + (x[, 2] + 1)^2) + 0.2 * rnorm(200)
fit <- ppm(x, y, loss = "lssvm", lambda = 0.01)
summary(fit, d = 2)
```

---

wdbc *Wisconsin Diagnostic Breast Cancer (WDBC) Data*

Description

Diagnostic measurements for 569 breast masses from the Wisconsin Diagnostic Breast Cancer study (Street, Wolberg and Mangasarian; 1993). Each record has thirty real-valued features computed from a digitized image of a fine-needle aspirate, namely the mean, standard error (`_se`) and worst (largest) value of ten cell-nucleus characteristics, together with a benign/malignant diagnosis. Features are on their natural scale; standardize them before fitting (see Examples).

Usage

```
wdbc
```

Format

A data frame with 569 rows and 31 variables: a factor `diagnosis` with levels `"B"` (benign, 357 cases) and `"M"` (malignant, 212 cases), followed by thirty numeric features. The features are the `_mean`, `_se` and `_worst` summaries of: `radius`, `texture`, `perimeter`, `area`, `smoothness`, `compactness`, `concavity`, `concave_points`, `symmetry` and `fractal_dimension` (for example `radius_mean`, `radius_se`, `radius_worst`).

Details

For the weighted-loss principal machines (`"wsvm"`, `"wlssvm"`, `"wlogit"`, `"wl2svm"`) the response is coded as malignant = `+1` and benign = `-1`, as shown in the Examples.

Source

Street, W. N., Wolberg, W. H. and Mangasarian, O. L. (1993) Nuclear feature extraction for breast tumor diagnosis. *Biomedical Image Processing and Biomedical Visualization*, 1905, 861–870. UCI Machine Learning Repository, Breast Cancer Wisconsin (Diagnostic) Data Set.

Examples

```
data(wdbc)
x <- scale(as.matrix(wdbc[, -1]))
y <- ifelse(wdbc$diagnosis == "M", 1, -1)
fit <- ppm(x, y, loss = "wl2svm", penalty = "grSCAD", lambda = 0.3)
summary(fit, d = 2)
B      <- fit$evectors[, 1:2]
scores <- x %*% B
plot(scores[, 1], scores[, 2], col  = ifelse(y == 1, "red", "blue"),
 pch  = ifelse(y == 1, 17, 1),
 xlab = "1st SDR direction", ylab = "2nd SDR direction")
legend("topright", legend = c("malignant (+1)", "benign (-1)"),
 col = c("red", "blue"), pch = c(17, 1))
```

Index

∗ **datasets**
boston, 2
wdbc, 13

boston, 2

message(), *10*

ppm, 4
ppm(), *10–13*
ppm_tune, 8
ppm_tune(), *12*
print.ppm, 11
print.ppm(), *7*, *13*
print.ppm_tune, 12

set.seed(), *9*
summary.ppm, 12
summary.ppm(), *7*, *11*

wdbc, 13