Sparse Sufficient Dimension Reduction via Penalized Principal Machines

Jungmin Shin (The Ohio State University) Seung Jun Shin (Korea University)

2026-06-13

Contents

1 Introduction

Sufficient dimension reduction (SDR) reduces the dimensionality of predictors $\mathbf{X}$ while preserving their relationship with a response Y. SDR estimates a basis $\mathbf{B}$ of the *central subspace* $\mathcal{S}_{Y|\mathbf{X}}$ defined by

$$Y \perp \mathbf{X} \mid \mathbf{B}^\top \mathbf{X}.$$

In high dimensions, **sparse SDR** improves interpretability and accuracy by driving many rows of $\mathbf{B}$ to zero, which is achieved by adding a sparsity-inducing penalty to the SDR optimization problem.

The **ppmSDR** package implements a unified framework for sparse SDR based on the penalized principal machine (P^2M). A single front-end, `ppm()`, dispatches to ten loss-specific estimators, all fitted by one group coordinate descent (GCD) engine, and `ppm_tune()` selects the sparsity parameter by cross-validation.

2 Penalized Principal Machines

2.1 Principal machine

Given data $(y_i, \mathbf{x}_i) \in \mathbb{R} \times \mathbb{R}^p$ with centered predictors, and a sequence of cutoffs $r_1 < \cdots < r_h$, the sample principal machine solves

$$(\beta_{0k}, \boldsymbol{\beta}_k) = \arg\min_{\beta_0, \boldsymbol{\beta}} \boldsymbol{\beta}^\top \hat{\Sigma} \boldsymbol{\beta} + \frac{c}{n} \sum_{i=1}^{n} L_k\Big(\tilde{y}_{ik}, \beta_0 + \boldsymbol{\beta}^\top \mathbf{x}_i\Big), \quad k = 1, \ldots, h,$$

where $\hat{\Sigma} = \sum_i \mathbf{x}_i \mathbf{x}_i^\top / n$. The basis $\hat{\mathbf{B}}$ is estimated by the leading d eigenvectors of $\sum_{k=1}^{h} \hat{\boldsymbol{\beta}}_k \hat{\boldsymbol{\beta}}_k^\top$.

- For the **response-based PM (RPM)**, $\tilde{y}_{ik} = \mathbb{I}\{y_i \geq r_k\} - \mathbb{I}\{y_i < r_k\}$ and the loss L_k is fixed across cutoffs.
- For the **loss-based PM (LPM)**, the loss L_k varies with r_k while $\tilde{y}_{ik}$ is fixed at y_i.

2.2 Penalized estimation

Under the sparsity assumption the slopes $\boldsymbol{\beta}_k$ share a common support across k, leading to the row-group penalized objective

$$\sum_{k=1}^{h} \left[\boldsymbol{\beta}_k^\top \hat{\Sigma} \boldsymbol{\beta}_k + \frac{c}{n} \sum_{i=1}^{n} L_k(\tilde{y}_{ik}, \beta_{0k} + \boldsymbol{\beta}_k^\top \mathbf{x}_i) \right] + \sum_{j=1}^{p} p_\lambda\Big(\|\boldsymbol{\beta}_{(j)}\|_2\Big),$$

where $\boldsymbol{\beta}_{(j)} = (\beta_{1j}, \ldots, \beta_{hj})^\top$ and $p_\lambda(\cdot)$ is the group LASSO, SCAD or MCP penalty. The penalty acts group-wise on all coefficients of predictor j, so variable selection corresponds to identifying the predictors that form a sparse basis.

2.3 Supported losses and algorithms

Machine	Response	Type	Loss $L_k(\tilde{y}_k, f)$	Algorithm
P^{2}LSM	Continuous	RPM	$(1 - \tilde{y}_k f)^2$	GCD
P^{2}WLSM	Binary	LPM	$w_k(1 - yf)^2$	GCD
P^{2}LR	Continuous	RPM	$\log(1 + e^{-\tilde{y}_k f})$	Iterative GCD
P^{2}WLR	Binary	LPM	$w_k \log(1 + e^{-yf})$	Iterative GCD
P^{2}AR	Both	LPM	$(y - f)^2(\rho_k \mathbb{I}\{y \geq f\} + (1 - \rho_k)\mathbb{I}\{y < f\})$	Iterative GCD
P^{2}L2M	Continuous	RPM	$[\max\{0, 1 - \tilde{y}_k f\}]^2$	Iterative GCD
P^{2}WL2M	Binary	LPM	$w_k[\max\{0, 1 - yf\}]^2$	Iterative GCD
P^{2}SVM	Continuous	RPM	$\max\{0, 1 - \tilde{y}_k f\}$	MM-GCD
P^{2}WSVM	Binary	LPM	$w_k \max\{0, 1 - yf\}$	MM-GCD
P^{2}QR	Both	LPM	$(y - f)(\rho_k - \mathbb{I}\{y < f\})$	MM-GCD

The squared loss yields an exact group-penalized least-squares problem solved directly by GCD. Smooth losses (logistic, asymmetric least squares, L2-hinge) are reduced to a sequence of penalized least-squares problems via quadratic approximation (iterative GCD). Non-differentiable losses (hinge, check) are handled by quadratic majorization within a majorization-minimization scheme (MM-GCD).

3 Usage

The argument `loss` selects the estimator; `penalty` is one of `"grSCAD"`, `"grLasso"` or `"grMCP"`; `lambda` controls sparsity and is best chosen by cross-validation (see below).

3.1 Regression

```
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 (P^2LSM)
fit <- ppm(x, y, H = 10, C = 1, loss = "lssvm", penalty = "grSCAD", lambda = 0.01)
round(fit$evectors[, 1:2], 3)
#>       [,1] [,2]
#>  [1,]    1    0
#>  [2,]    0    1
#>  [3,]    0    0
#>  [4,]    0    0
#>  [5,]    0    0
#>  [6,]    0    0
#>  [7,]    0    0
#>  [8,]    0    0
#>  [9,]    0    0
#> [10,]    0    0
summary(fit, d = 2)
#> Penalized Principal Machine (P2M) summary
#> Loss: lssvm   Penalty: grSCAD   lambda = 0.01
#> Working dimension d = 2
#>
#> Estimated basis of the central subspace:
#>      Dir1   Dir2
#> x1  1e+00 -1e-04
#> x2  1e-04  1e+00
#> x3  0e+00  0e+00
#> x4  0e+00  0e+00
#> x5  0e+00  0e+00
#> x6  0e+00  0e+00
#> x7  0e+00  0e+00
#> x8  0e+00  0e+00
#> x9  0e+00  0e+00
#> x10 0e+00  0e+00
#>
#> Selected variables (2 of 10): x1, x2
```

3.2 Binary classification

For a binary response the weighted losses code the classes internally as $\{-1, +1\}$.

```r
y.binary <- sign(y)
## penalized principal weighted least-squares SVM (P^2WLSM)
fit2 <- ppm(x, y.binary, H = 10, C = 1, loss = "asls",
            penalty = "grSCAD", lambda = 0.03)
round(fit2$evectors[, 1:2], 3)
#>        [,1]   [,2]
#>  [1,] 1.000 -0.005
#>  [2,] 0.005  1.000
#>  [3,] 0.000  0.000
#>  [4,] 0.000  0.000
#>  [5,] 0.005  0.004
#>  [6,] 0.000  0.000
#>  [7,] 0.006  0.001
#>  [8,] 0.000  0.000
#>  [9,] 0.000  0.000
#> [10,] 0.000  0.000
```

3.3 Choosing `lambda` by cross-validation

`ppm_tune()` performs K-fold cross-validation, selecting the `lambda` that maximizes the held-out distance correlation between the response and the estimated sufficient predictors. Call `set.seed()` beforehand for reproducible folds.

```r
set.seed(1)
cv <- ppm_tune(x, y, loss = "lssvm", d = 2, n.fold = 5,
               nlambda = 10, lambda.max = 0.02)
cv$opt.lambda
#> [1] 0.009283178
summary(cv$fit, d = 2)
#> Penalized Principal Machine (P2M) summary
#> Loss: lssvm   Penalty: grSCAD   lambda = 0.00928318
#> Working dimension d = 2
#>
#> Estimated basis of the central subspace:
#>       Dir1   Dir2
#> x1  1e+00 -3e-04
#> x2  3e-04  1e+00
#> x3  0e+00  0e+00
#> x4  0e+00  0e+00
#> x5  0e+00  0e+00
#> x6  0e+00  0e+00
#> x7  0e+00  0e+00
#> x8  0e+00  0e+00
#> x9  0e+00  0e+00
#> x10 0e+00  0e+00
#>
#> Selected variables (2 of 10): x1, x2
```

4 Real-data example: Wisconsin Diagnostic Breast Cancer

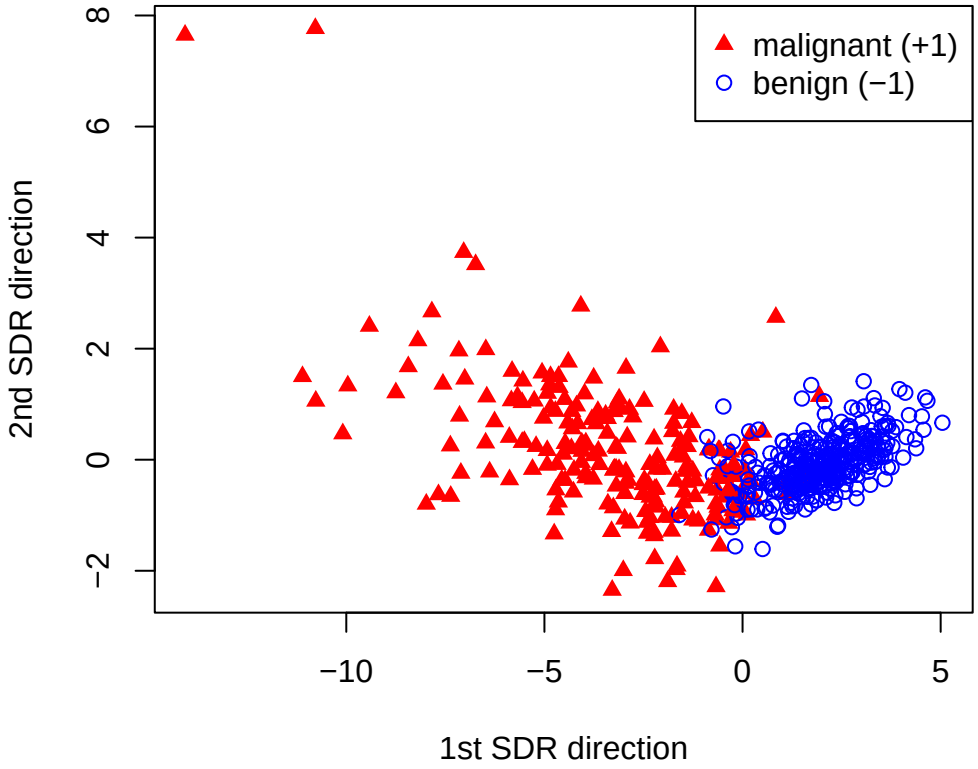
We illustrate a weighted estimator on the Wisconsin Diagnostic Breast Cancer (WDBC) data, coding the diagnosis as malignant = +1 and benign = −1. After fitting the penalized principal weighted L2-SVM (P2WL2M), the predictors are projected onto the estimated two-dimensional central subspace and plotted by class.

```
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)
#> Penalized Principal Machine (P2M) summary
#> Loss: wl2svm   Penalty: grSCAD   lambda = 0.3
#> Working dimension d = 2
#>
#> Estimated basis of the central subspace:
#>                            Dir1    Dir2
#> radius_mean             -0.2733 -0.3393
#> texture_mean             0.0000  0.0000
#> perimeter_mean          -0.2929 -0.3362
#> area_mean               -0.2650  0.0599
#> smoothness_mean          0.0000  0.0000
#> compactness_mean        -0.0819 -0.0947
#> concavity_mean          -0.2452  0.0401
#> concave_points_mean     -0.3706  0.2268
#> symmetry_mean            0.0000  0.0000
#> fractal_dimension_mean   0.0000  0.0000
#> radius_se               -0.1022  0.3825
#> texture_se               0.0000  0.0000
#> perimeter_se            -0.0773  0.2433
#> area_se                 -0.0670  0.2180
#> smoothness_se            0.0000  0.0000
#> compactness_se           0.0000  0.0000
#> concavity_se             0.0000  0.0000
#> concave_points_se        0.0000  0.0000
#> symmetry_se              0.0000  0.0000
#> fractal_dimension_se     0.0000  0.0000
#> radius_worst            -0.3641  0.0832
#> texture_worst            0.0000  0.0000
#> perimeter_worst         -0.3715  0.0384
#> area_worst              -0.3223  0.4892
#> smoothness_worst         0.0000  0.0000
#> compactness_worst       -0.0743 -0.0829
#> concavity_worst         -0.1698 -0.2341
#> concave_points_worst    -0.3679 -0.3794
#> symmetry_worst           0.0000  0.0000
#> fractal_dimension_worst  0.0000  0.0000
#>
#> Selected variables (15 of 30): radius_mean, perimeter_mean, area_mean, compactness_mean, concavity_m

B      <- fit$evectors[, 1:2]
scores <- x %*% B
```

```r
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))
```



The Boston housing data (`data(boston)`) provide a continuous-response counterpart for the response-based estimators such as `"lssvm"`.

5 References

- Li, B., Artemiou, A. and Li, L. (2011). Principal support vector machines for linear and nonlinear sufficient dimension reduction. *Annals of Statistics*, 39(6), 3182–3210.
- Shin, S. J. and Artemiou, A. (2017). Penalized principal logistic regression for sparse sufficient dimension reduction. *Computational Statistics & Data Analysis*, 111, 48–58.
- 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.
- Shin, J., Shin, S. J. and Artemiou, A. (2024). The R package psvmSDR: a unified algorithm for sufficient dimension reduction via principal machines. *arXiv:2409.01547*.