Sparse Sufficient Dimension Reduction via Penalized Principal Machines

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

2026-09-23

Contents

1 Introduction **1**

2 Penalized Principal Machines **2**

2.1 Principal machine 2

2.2 Penalized estimation 2

2.3 Supported losses and algorithms 2

2.4 Step-halving line search 3

3 Usage **3**

3.1 Regression 3

3.2 Binary classification 4

3.3 Monitoring convergence with the line search 5

3.4 Choosing `lambda` by cross-validation 6

4 Real-data example: Wisconsin Diagnostic Breast Cancer **7**

5 References **9**

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

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

where $\boldsymbol{\theta}$ stacks the intercepts and slopes of the h machines, $\boldsymbol{\beta}_{(j)} = (\beta_{1j}, \ldots, \beta_{hj})^\top$ 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	`line.search` default
P^{2}LSM	Continuous	RPM	$(1 - \tilde{y}_k f)^2$	GCD	not applicable
P^{2}WLSM	Binary	LPM	$w_k(1 - yf)^2$	GCD	not applicable
P^{2}LR	Continuous	RPM	$\log(1 + e^{-\tilde{y}_k f})$	Iterative GCD	`TRUE`
P^{2}WLR	Binary	LPM	$w_k \log(1 + e^{-yf})$	Iterative GCD	`TRUE`
P^{2}AR	Both	LPM	$(y - f)^2(\tau_k \mathbb{I}\{y \geq f\} + (1 - \tau_k)\mathbb{I}\{y < f\})$	Iterative GCD	`FALSE`
P^{2}L2M	Continuous	RPM	$[\max\{0, 1 - \tilde{y}_k f\}]^2$	Iterative GCD	`TRUE`
P^{2}WL2M	Binary	LPM	$w_k[\max\{0, 1 - yf\}]^2$	Iterative GCD	`TRUE`
P^{2}SVM	Continuous	RPM	$\max\{0, 1 - \tilde{y}_k f\}$	MM-GCD	`FALSE`
P^{2}WSVM	Binary	LPM	$w_k \max\{0, 1 - yf\}$	MM-GCD	`FALSE`
P^{2}QR	Both	LPM	$(y - f)(\tau_k - \mathbb{I}\{y < f\})$	MM-GCD	`FALSE`

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).

2.4 Step-halving line search

The quadratic approximations used by the iterative GCD algorithm are local approximations rather than majorizers of the loss, so a plain GCD update does not by itself guarantee that $Q(\boldsymbol{\theta})$ decreases. With `line.search = TRUE`, the candidate $\hat{\boldsymbol{\theta}}^{(t)}$ returned by the GCD step at iteration t is accepted in the damped form

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

where $m_t \geq 0$ is the smallest integer for which $Q(\boldsymbol{\theta}^{(t+1)}) \leq Q(\boldsymbol{\theta}^{(t)})$, evaluated on the penalized objective itself rather than on the quadratic surrogate. If no such $m_t \leq$ `max.halving` (default 10) exists, the current iterate is retained and the algorithm stops. The safeguard therefore guarantees monotone non-increase of $Q(\boldsymbol{\theta})$, and it coincides with the plain update ($m_t = 0$) whenever the plain update does not increase the objective. The default `line.search = NULL` selects the loss-specific default in the table above, and `line.search = FALSE` always gives the plain updates.

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
```

The same interface fits any other estimator. For example, the penalized principal asymmetric least squares regression (P^2AR) examines the response at several asymmetry levels τ_k:

```
## penalized principal asymmetric least squares regression (P^2AR)
fit_ar <- ppm(x, y, H = 10, C = 1, loss = "asls", penalty = "grSCAD", lambda = 0.05)
round(fit_ar$evectors[, 1:2], 3)
#>       [,1]  [,2]
#>  [1,] 1.00 -0.02
#>  [2,] 0.02  1.00
#>  [3,] 0.00  0.00
#>  [4,] 0.00  0.00
#>  [5,] 0.00  0.00
#>  [6,] 0.00  0.00
#>  [7,] 0.00  0.00
#>  [8,] 0.00  0.00
#>  [9,] 0.00  0.00
#> [10,] 0.00  0.00
```

3.2 Binary classification

For a binary response the weighted losses (`"wlssvm"`, `"wlogit"`, `"wl2svm"`, `"wsvm"`) code the classes internally as $\{-1, +1\}$, and P^2AR applies to both response types. The example below generates a binary response whose central subspace is again spanned by the first two predictors.

```
yb <- sign(x[, 1] + x[, 2]^3 / 3 + 0.2 * rnorm(n))

## penalized principal asymmetric least squares (P^2AR)
fit_arb <- ppm(x, yb, loss = "asls", penalty = "grSCAD", lambda = 0.2)
round(fit_arb$evectors[, 1:2], 3)
#>        [,1]   [,2]
#>  [1,] 0.927 -0.376
#>  [2,] 0.376  0.927
#>  [3,] 0.000  0.000
#>  [4,] 0.000  0.000
#>  [5,] 0.000  0.000
#>  [6,] 0.000  0.000
#>  [7,] 0.000  0.000
#>  [8,] 0.000  0.000
#>  [9,] 0.000  0.000
#> [10,] 0.000  0.000
```

```r
## penalized principal weighted logistic regression (P^2WLR)
fit_wlr <- ppm(x, yb, loss = "wlogit", penalty = "grSCAD", lambda = 0.005)
summary(fit_wlr, d = 2)
#> Penalized Principal Machine (P2M) summary
#> Loss: wlogit   Penalty: grSCAD   lambda = 0.005
#> Algorithm: iterative GCD (line search on), 7 iterations, converged
#> Working dimension d = 2
#>
#> Estimated basis of the central subspace:
#>       Dir1    Dir2
#> x1  0.9450 -0.3271
#> x2  0.3271  0.9450
#> x3  0.0000  0.0000
#> x4  0.0000  0.0000
#> x5  0.0000  0.0000
#> x6  0.0000  0.0000
#> x7  0.0000  0.0000
#> x8  0.0000  0.0000
#> x9  0.0000  0.0000
#> x10 0.0000  0.0000
#>
#> Selected variables (2 of 10): x1, x2
```

3.3 Monitoring convergence with the line search

For the iterative GCD losses the fitted object records the number of outer iterations (`iter`), the termination status (`status`), and, when the line search is on, the objective path (`objective`) and the number of halvings m_t at each iteration (`n.halving`). For P^{2}LR the safeguard is on by default:

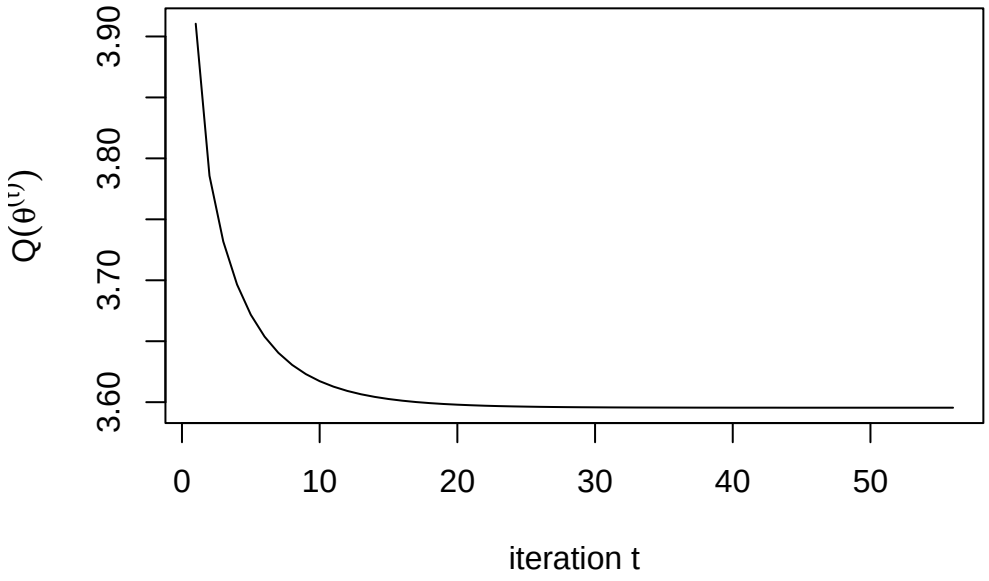
```r
## penalized principal logistic regression (P^2LR)
fit_lr <- ppm(x, y, loss = "logit", penalty = "grSCAD", lambda = 0.01)
print(fit_lr)
#> Penalized Principal Machine (P2M) for Sufficient Dimension Reduction
#>
#> Call:
#>   ppm(x = x, y = y, loss = "logit", lambda = 0.01, penalty = "grSCAD")
#>
#> Loss: logit   Penalty: grSCAD   lambda = 0.01   gamma = 3.7   C = 1   H = 10
#> Data: n = 1000, p = 10 (continuous response)
#> Algorithm: iterative GCD, step-halving line search on (max.halving = 10; 0 of 7 updates damped)
#> Iterations: 7 (converged)
#> Leading eigenvalues of the working matrix M:
#>   0.00040482, 5.05e-06, 0, 0, 0
fit_lr$n.halving                  # m_t at each iteration (0 = plain update)
#> [1] 0 0 0 0 0 0 0
all(diff(fit_lr$objective) <= 0)  # Q never increases
#> [1] TRUE

## the plain updates give the same fit whenever no step is damped
fit_lr0 <- ppm(x, y, loss = "logit", penalty = "grSCAD", lambda = 0.01,
               line.search = FALSE)
all.equal(fit_lr$M, fit_lr0$M)
```

```
#> [1] TRUE
```

For P^{2}AR the plain updates are the default, and the safeguard can be requested explicitly. Here it damps an update, terminates earlier, and returns the same basis as the plain updates.

```
fit_ar_ls <- ppm(x, y, loss = "asls", penalty = "grSCAD", lambda = 0.05,
                 line.search = TRUE)
c(plain = fit_ar$iter, safeguarded = fit_ar_ls$iter)
#>       plain safeguarded
#>          88          56
table(fit_ar_ls$n.halving)        # how often each m_t was used
#>
#>  0  5
#> 55  1
round(fit_ar_ls$evectors[, 1:2], 3)
#>       [,1]  [,2]
#>  [1,] 1.00 -0.02
#>  [2,] 0.02  1.00
#>  [3,] 0.00  0.00
#>  [4,] 0.00  0.00
#>  [5,] 0.00  0.00
#>  [6,] 0.00  0.00
#>  [7,] 0.00  0.00
#>  [8,] 0.00  0.00
#>  [9,] 0.00  0.00
#> [10,] 0.00  0.00
plot(fit_ar_ls$objective, type = "l",
     xlab = "iteration t", ylab = expression(Q(theta^(t))))
```



3.4 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. The line-search settings are passed to every cross-validation fit.

```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

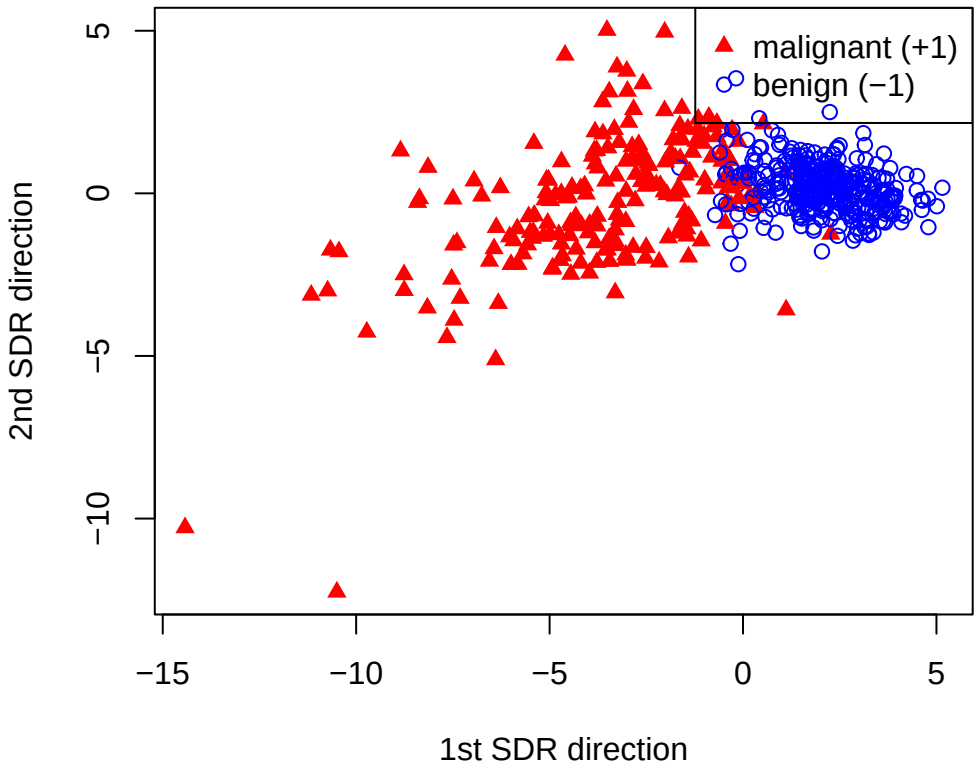
We illustrate a weighted estimator on the Wisconsin Diagnostic Breast Cancer (WDBC) data, coding the diagnosis as malignant = +1 and benign = −1. After fitting the penalized principal weighted L2-SVM (P2WL2M), the predictors are projected onto the estimated two-dimensional central subspace and plotted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

```r
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
#> Algorithm: iterative GCD (line search on), 7 iterations, line.search.halted
#> Working dimension d = 2
#>
#> Estimated basis of the central subspace:
#>                            Dir1    Dir2
#> radius_mean             -0.2726  0.0026
#> texture_mean            -0.0669  0.0658
#> perimeter_mean          -0.2817 -0.0188
#> area_mean               -0.2440 -0.2586
#> smoothness_mean          0.0000  0.0000
#> compactness_mean        -0.1299  0.0219
#> concavity_mean          -0.2374 -0.0825
```

```
#> concave_points_mean     -0.3334 -0.0962
#> symmetry_mean            0.0000  0.0000
#> fractal_dimension_mean   0.0000  0.0000
#> radius_se               -0.1270 -0.3234
#> texture_se               0.0000  0.0000
#> perimeter_se            -0.1005 -0.2788
#> area_se                 -0.0867 -0.2955
#> 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            -0.3386 -0.0575
#> texture_worst           -0.1321  0.2224
#> perimeter_worst         -0.3376 -0.0785
#> area_worst              -0.2830 -0.3654
#> smoothness_worst        -0.0868  0.1131
#> compactness_worst       -0.1708  0.2474
#> concavity_worst         -0.2336  0.3465
#> concave_points_worst    -0.3757  0.4956
#> symmetry_worst          -0.0752  0.1008
#> fractal_dimension_worst  0.0000  0.0000
#>
#> Selected variables (19 of 30): radius_mean, texture_mean, perimeter_mean,
#>   area_mean, compactness_mean, concavity_mean, concave_points_mean, radius_se,
#>   perimeter_se, area_se, radius_worst, texture_worst, perimeter_worst,
#>   area_worst, smoothness_worst, compactness_worst, concavity_worst,
#>   concave_points_worst, symmetry_worst

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))
```



If the summary reports `line.search.halted`, the safeguard stopped at the last iterate that decreased the penalized objective, because a further GCD update would have increased it; `fit$iter` and `fit$n.halving` show where this happened. Refitting with `line.search = FALSE` gives the plain updates.

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*.