TP 10 MLA : MCP

Stéphane Canu

23 novembre 2024, ITI, INSA Rouen

Le but du TP est d'étudier une autre méthode de sélection de variables, le *minimax concave penalties* (MCP), dans le cadre de la régression sur des données partiellement réelles. Pour le faire fonctionner, vous êtes supposé avoir déjà installé CVX.

Ex. 1 — MCP et CVX

On suppose connues la matrice $X \in \mathbb{R}^{n\times p}$ et le vecteur $y \in \mathbb{R}^n$.

1. Ecrire un programme python permettant de résoudre avec CVX le problème de minimisation de J_{MCP}, le cout des moindres carrés pénalisé quand la pénalité est de type MCP (on pourra utiliser la fonction cvxpy `cvx.minimum(.,.)`)

$$J_{\text{MCP}}(\beta) \;=\; \tfrac{1}{2}\|y - X\beta\|^2 + \lambda \sum_{j=1}^{p} \mathbf{pen}_{\lambda,\gamma}(|\beta_j|)\;,$$

où $\mathbf{pen}_{\lambda,\gamma}$ est la pénalité du MCP définie par, pour λ et γ donnés, :

$$\mathbf{pen}_{\lambda,\gamma}(t) = \int_0^t \left(1 - \frac{x}{\gamma\lambda}\right)_+ dx = \begin{cases} t - \dfrac{t^2}{2\lambda\gamma} & \text{if } t \le \gamma\lambda \\ \dfrac{\gamma\lambda}{2} & \text{else.} \end{cases}$$

Vous devriez avoir rencontré un problème. Expliquez la nature de la difficultée rencontrée.

Ex. 2 — le cout le gradient et les condition d'optimalité du MCP

1. Ecrire une fonction python `cout_mcp` permettant de calculer le cout MCP d'un problème de régression

```
def cout_MCP(X,y,b,lam, gam):
  e = ...
  p = np.ones(np.shape(b))*gam*lam/2
  ind = np.where(np.abs(b) < gam*lam)
  if ind[0].shape[0]> 0:
    p[ind] = ...
  return ...
```

2. Ecrire une fonction python `verif_grad_mcp` permettant de vérifier qu'un β est bien optimal, en vérifiant que $0 \in \partial_\beta J_{MCP}$ avec, pour $\alpha_j \in [-1, 1]$,

$$\partial_c J_{\text{MCP}}(\beta) \;=\; X^\top(X\beta - y) + \lambda \begin{cases} \alpha_j & \text{if } \beta_j = 0 \\ \text{sign}(\beta_j) - \dfrac{\beta_j}{\lambda\gamma} & \text{if } |\beta_j| \le \lambda\gamma \\ 0 & \text{else}\;, \end{cases} \tag{1}$$

```
def verif_grad_MCP(X,y,b,lam, gam):
  g = ...
    ...
  ind = np.where(np.abs(b) < gam*lam)
  if ind[0].shape[0]> 0:
    p[ind] = ...
  return ...
```

3. Proposez une fonction permettant de visualiser la pénalité MCP

Ex. 3 — le solveur component wise du MCP

1. La solution monovariable du MCP est trouvé à partir de la sous différentielle :

$$\frac{\partial J_{\text{MCP}}(\beta)}{\partial \beta_j} = X_{\bullet j}^\top (X\beta - \mathbf{r}) + \begin{cases} \lambda \alpha_j & \text{if } |\beta_j| = 0 \\ sign(\beta_j) \max\left(0, \lambda - \frac{|\beta_j|}{\gamma}\right) & \text{else,} \end{cases}$$

avec $\alpha_j \in [-1, 1]$. Le vecteur $X_{\bullet j}$ denote la colonne j de la matrice X. Dans ce cas, en supposant $X_{\bullet j}^\top X_{\bullet j} = 1$,

$$0 \in \frac{\partial J_{\text{MCP}}(\beta)}{\beta_j} \quad \Leftrightarrow \quad \beta_j = \begin{cases} 0 & \text{if } |X_{\bullet j}^\top \mathbf{r}| \leq \lambda \\ s_j \left(|X_{\bullet j}^\top \mathbf{r}| - \lambda\right) \dfrac{\gamma}{\gamma - 1} & \text{if } |X_{\bullet j}^\top \mathbf{r}| \leq \lambda\gamma \\ X_{\bullet j}^\top \mathbf{r} & \text{sinon,} \end{cases}$$

avec s_j le signe de $X_{\bullet j}^\top \mathbf{r}$ et $\mathbf{r} = X\beta - y - X_{\bullet j}\beta_j$ l'erreur résiduelle.

a) Adaptez l'algorithme "component wise" pour le lasso à la pénalité MCP et proposez une fonction donnant l'estimateur minimisant le cout MCP à partir de kla matrice X, du vecteur y et des paramètres λ et γ

```
def MCP_CW( X,y, lambda, gam, beta_init):
  ...
  return beta
```

b) Ecrire une fonction permettent de tester votre fonction `MCP_CW`. Vérifiez que le cout diminue et que la solution vérifie bien les conditions d'optimalisés (Vous pourrez utiliser le problème jouet du dernier TD avec le même λ que pour le lasso et $\gamma = 5$.

Ex. 4 — le solveur DC du MCP

1. Based on this decomposition, the DC algorithm amounts to iteratively building a sequence by minimizing, for a given β^{old}, the following convex surrogate cost function

$$J_{\text{DC}}(\beta) = \tfrac{1}{2}\|y - X\beta\|^2 + \lambda\|\beta\|_1 - \lambda \sum_{j=1}^{p} h'_{\lambda,\gamma}(|\beta_j^{old}|)\, |\beta_j|, \tag{2}$$

h' being the derivative of the Huber loss function. It turns out that in that case minimizing the DC cost function (2) can be seen as minimizing an adaptive Lasso.

$$\min_{\beta \in \mathbf{R}^p} \ \tfrac{1}{2}\|y - X\beta\|^2 + \lambda \sum_{j=1}^{p} w_j |\beta_j|, \tag{3}$$

avec les poids

$$w_j = \begin{cases} 0 & \text{if } |\beta_j^{old}| > \lambda\gamma \\ 1 - \dfrac{|\beta_j^{old}|}{\gamma\lambda} & \text{else} \end{cases}.$$

a) Ecrire une fonction matlab `MCP_DC` permettant de résoudre le MCP à l'aide d'une méthode de type DC.

b) Ecrire une fonction permettent de tester votre fonction `MCP_DC`. Vérifiez que le cout diminue et que la solution vérifie bien les conditions d'optimalisés

Ex. 5 — Comparaison des solveurs CW et DC du MCP

1. comparez les solutions des deux solveurs sur un problème jouet.
2. comparez la solution du MCP sur les données "prostate" du premier TP, avec par exemple $\lambda = 5$ et $\gamma = 2$.