

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 (que vous pourrez télécharger à cette adresse : <http://cvxr.com/cvx/>)

Ex. 1 — 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

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

où $\text{pen}_{\lambda,\gamma}$ est la pénalité du MCP définie par :

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

```
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 $0 \in \partial_{\beta} J_{\text{MCP}}$ avec

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

et $\alpha_j \in [-1, 1]$.

```
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. 2 — le solveur component wise du MCP

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

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

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

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

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 la 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 permettant 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. 3 — 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

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

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.

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2} \|y - X\beta\|^2 + \lambda \sum_{j=1}^p w_j |\beta_j|, \quad (3)$$

avec les poids

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

- 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 permettant de tester votre fonction `MCP_DC`. Vérifiez que le cout diminue et que la solution vérifie bien les conditions d'optimalisés

Ex. 4 — Comparaison des solveurs CW et DC du MCP

1. comparez les solutions des deux solveurs sur le problème jouet du TP 4.