Optimisation bi-niveaux pour l'apprentissage d'hyperparamètres

Jordan Frécon

Equipe Apprentissage - LITIS - INSA de Rouen

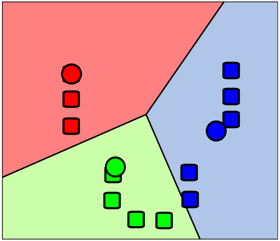
Sommaire

Introduction

Qu'est-ce qu'un hyperparamètre ?

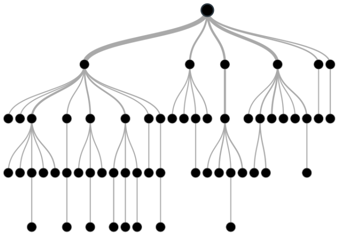
Hyperparamètre

Un hyperparamètre λ est un paramètre défini avant le début du processus d'apprentissage. Ces (hyper)-paramètres sont réglables et peuvent affecter directement les performances d'un modèle et la façon dont il s'apprend.



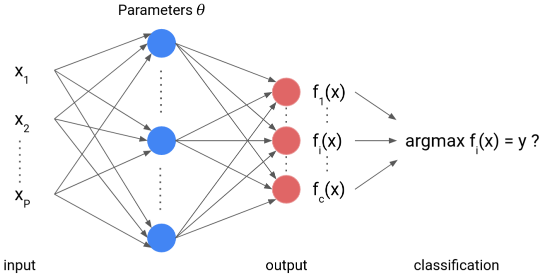
Hyperparamètre λ = nombre de clusters k

Exemple 2 : Arbre de décision



Hyperparamètre λ = nombre de branches

Exemple 3 : Réseaux de neurones



Hyperparamètre λ = nombre d'epochs, taux d'apprentissage, ...

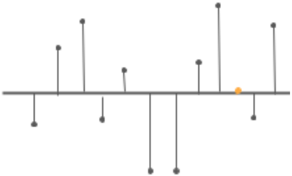
Exemple 4 : Lasso

Soient des données $\{x_i, y_i\}_{i=1}^n$, on souhaite trouver un **modèle de régression linéaire parcimonieux** entre x et y.

En d'autres termes, on cherche w parcimonieux de telle sorte que $y_i \approx x_i w$.

$$\min_w \left\{ \mathcal{L}(w; \lambda) \triangleq \frac{1}{2} \|y - Xw\|^2 + \lambda \|w\|_1 \right\}$$

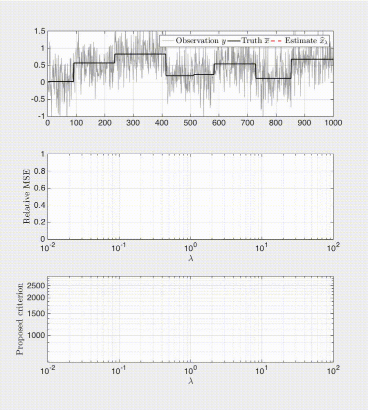
λ faible



λ élevé

Hyperparamètre λ = paramètre de régularisation

$$\min_{w} \left\{ \mathcal{L}(w;\lambda) \triangleq \frac{1}{2}\|y - w\|^2 + \lambda \mathrm{TV}(w) \right\}$$



Hyperparamètre λ = paramètre de régularisation

Entiers

- nombre de clusters dans un algorithme de partitionnement
- nombre de branches dans un arbre de decision
- nombre d'epochs

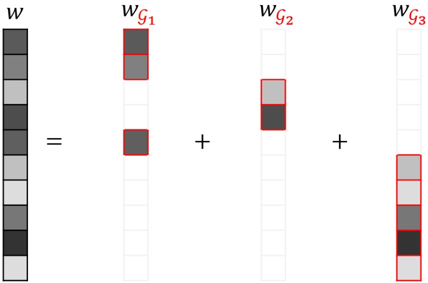
Réels

- taux d'apprentissage
- moments
- paramètres de régularisation

Particularité mon domaine de recherche : **Structures** (groupes, hiérarchies, graphes)

La solution du *Group Lasso* [Yuan and Lin (2006)] dépend des groupes $\{\mathcal{G}_1, \dots, \mathcal{G}_L\}$

$$\hat{w}(\mathcal{G}_1, \dots, \mathcal{G}_L) = \operatorname*{argmin}_{w} \frac{1}{2}\|y - Xw\|^2 + \gamma \sum_{l=1}^{L} \|w_{\mathcal{G}_l}\|_2$$



Hyperparamètre λ = Groupes $\{\mathcal{G}_1, \dots, \mathcal{G}_L\}$

Rappels sur la sélection de modèles

Pour chaque hyperparamètre λ, nous avons un modèle $\hat{w}(\lambda)$ différent

Exemples :

$\hat{w}(\lambda)$ = partitionnement en λ moyennes (k-means clustering)

$\hat{w}(\lambda)$ = réseau de neurones appris avec un taux d'apprentissage valant λ

Choisir $\lambda \Leftrightarrow$ Sélection de modèle $\{\hat{w}(\lambda)\}_\lambda$

Questions :

1. Dans quel espace choisir λ ?
2. Comment évaluer la qualité du modèle $\hat{w}(\lambda)$?

Sélection de modèles

Pour chaque hyperparamètre λ, nous avons un modèle $\hat{w}(\lambda)$ différent

Exemples :

$\hat{w}(\lambda)$ = partitionnement en λ moyennes (k-means clustering)

$\hat{w}(\lambda)$ = réseau de neurones appris avec un taux d'apprentissage valant λ

Choisir $\lambda \Leftrightarrow$ Sélection de modèle $\{\hat{w}(\lambda)\}_\lambda$

Questions :

1. Dans quel espace choisir λ ?
2. Comment évaluer la qualité du modèle $\hat{w}(\lambda)$?

Une stratégie populaire consiste à choisir λ sur une grille Λ prédéfinie

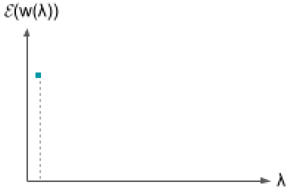
La qualité du modèle $\hat{w}(\lambda)$ est évaluée selon un critère $\mathcal{E}$:

- critère d'information d'Akaike (AIC)
- critère d'information bayésien (BIC)
- norme induisant la parcimonie
- $\vdots$
- erreur de validation

Pour chaque $\lambda \in \Lambda = \{\lambda_{\text{min}}, \ldots, \lambda_{\text{max}}\}$

- Calculer $\hat{w}(\lambda)$
- Evaluer $\mathcal{E}(\hat{w}(\lambda))$

Choisir $\hat{\lambda}$ qui minimise $\{\mathcal{E}(\hat{w}(\lambda))\}_{\lambda \in \Lambda}$



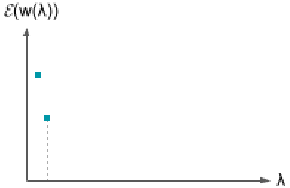
Principe de la recherche sur grille

Pour chaque $\lambda \in \Lambda = \{\lambda_{\mathsf{min}}, \ldots, \lambda_{\mathsf{max}}\}$

$\quad$ Calculer $\hat{w}(\lambda)$

$\quad$ Evaluer $\mathcal{E}(\hat{w}(\lambda))$

Choisir $\hat{\lambda}$ qui minimise $\{\mathcal{E}(\hat{w}(\lambda))\}_{\lambda \in \Lambda}$

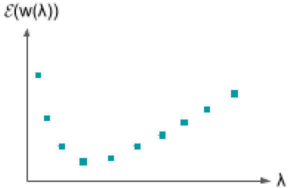


Pour chaque $\lambda \in \Lambda = \{\lambda_{\text{min}}, \ldots, \lambda_{\text{max}}\}$

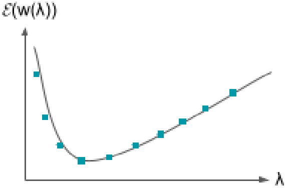
  Calculer $\hat{w}(\lambda)$

  Evaluer $\mathcal{E}(\hat{w}(\lambda))$

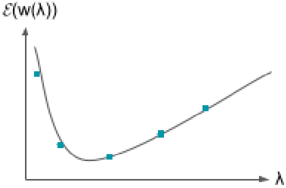
Choisir $\hat{\lambda}$ qui minimise $\{\mathcal{E}(\hat{w}(\lambda))\}_{\lambda\in\Lambda}$



Grille fine

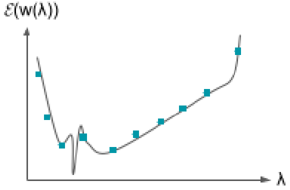


Grille grossière



La grille doit être choisie selon les propriétés de $\lambda \mapsto \mathcal{E}(\hat{w}(\lambda))$

Limitations de la recherche sur grille



La fonction $\lambda \mapsto \mathcal{E}(\hat{w}(\lambda))$ peut être non convexe

Formalisme bi-niveaux pour l'apprentissage d'hyperparamètres

Problème bi-niveaux

$$\min_{\lambda \in \Lambda} \mathcal{E}(\hat{w}(\lambda), \lambda) \qquad \text{s.c.} \qquad \hat{w}(\lambda) = \operatorname*{argmin}_{w} \mathcal{L}(w; \lambda)$$

Le critère $\mathcal{E}$ peut également dépendre de λ et l'espace de recherche Λ est un ensemble continu

Questions :

1. Est-ce que le problème est bien posé ?
2. Est-ce qu'il possède des solutions ?
3. Si oui, comment trouver ces solutions ?

Problème bi-niveaux

$$\min_{\lambda \in \Lambda} \mathcal{E}(\hat{w}(\lambda), \lambda) \qquad \text{s.c.} \qquad \hat{w}(\lambda) = \operatorname*{argmin}_{w} \mathcal{L}(w; \lambda)$$

Partitionnement en k-moyennes.
Soient des données $\{x_i\}_{i=1}^n$. On souhaite trouver $k = \lambda$ clusters, notés $w_1, \ldots, w_\lambda$, via la minimisation de

$$\mathcal{L}(w; \lambda) = \sum_{i=1}^{\lambda} \sum_{x \in w_i} \|x - \mu_i\|^2$$

où est la moyenne des points dans le cluster w_i est noté μ_i.

Problème NP difficile !

Problème bi-niveaux

$$\min_{\lambda \in \Lambda} \mathcal{E}(\hat{w}(\lambda), \lambda) \qquad \text{s.c.} \qquad \hat{w}(\lambda) = \operatorname*{argmin}_{w} \mathcal{L}(w; \lambda)$$

Probleme Lasso.
Soient des données $\{x_i, y_i\}_{i=1}^n$, on souhaite trouver un modèle de régression linéaire parcimonieux entre x et y. En d'autres termes, on cherche w parcimonieux de telle sorte que $y_i \approx x_i w$. Pour cela, on considère la minimisation de

$$\mathcal{L}(w; \lambda) = \frac{1}{2}\|y - xw\|^2 + \lambda\|w\|_1$$

L'objectif $\mathcal{L}(\cdot; \lambda)$ est convexe mais non différentiable

Problème bi-niveaux

$$\min_{\lambda \in \Lambda} \mathcal{E}(\hat{w}(\lambda), \lambda) \qquad \text{s.c.} \qquad \hat{w}(\lambda) = \operatorname*{argmin}_{w} \mathcal{L}(w; \lambda)$$

On souhaite :

- Qu'il existe une seule solution $\hat{w}(\lambda)$
- La solution $\hat{w}(\lambda)$ soit *facile* à trouver

$$\Rightarrow \mathcal{L}(\cdot; \lambda) \text{ strictement convexe}$$

Problème bi-niveaux

$$\min_{\lambda \in \Lambda} \mathcal{E}(\hat{w}(\lambda), \lambda) \qquad \text{s.c.} \qquad \hat{w}(\lambda) = \operatorname*{argmin}_{w} \mathcal{L}(w; \lambda)$$

Théorème informel

Si Λ est compacte, $\mathcal{L}(\cdot; \lambda)$ strictement convexe et $\mathcal{E}$ est continu, alors $\lambda \mapsto \hat{w}(\lambda)$ est continu et le problème bi-niveaux admet des solutions.

« J. Frecon, S. Salzo, and M. Pontil. Bilevel learning of the group lasso structure. In Advances in Neural Information Processing Systems 31 (NeurIPS), pages 8301–8311. 2018 »

Problème bi-niveaux

$$\min_{\lambda \in \Lambda} \mathcal{E}(\hat{w}(\lambda), \lambda) \qquad \text{s.c.} \qquad \hat{w}(\lambda) = \operatorname*{argmin}_{w} \mathcal{L}(w; \lambda)$$

- Recherche sur grille
- Recherche aléatoire
- Algorithmes évolutionnaires

  $\vdots$

- Algorithmes basés sur le gradient $\longrightarrow$ sujet de la suite de cet exposé !

Optimisation bi-niveaux basée sur le gradient

Problème bi-niveaux

$$\min_{\lambda \in \Lambda} \left\{ \mathcal{U}(\lambda) \triangleq \mathcal{E}(\hat{w}(\lambda), \lambda) \right\} \qquad \text{s.c.} \qquad \hat{w}(\lambda) = \operatorname*{argmin}_{w} \mathcal{L}(w; \lambda)$$

On peut voir le problème supérieur comme visant à minimiser un objectif $\mathcal{U}$

Choix initial $\lambda_0 \in \Lambda$

Pour $q = 0, \ldots,$

$\lfloor \; \lambda_{q+1} = \mathrm{Proj}_{\Lambda}\left(\lambda_q - \eta \nabla \mathcal{U}(\lambda_q)\right)$

Problèmes potentiels

- Cela suppose que l'on connaisse $\hat{w}(\lambda)$ de manière exacte !
- $\mathcal{U}$ n'est pas forcément différentiable
- Comment choisir le pas η ? Est-ce que $\mathcal{U}$ est localement lisse ?

Problème bi-niveaux

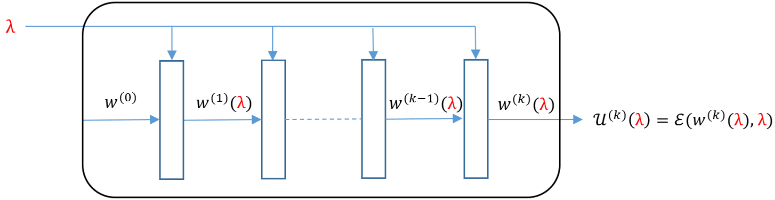
$$\min_{\lambda \in \Lambda} \left\{ \mathcal{U}(\lambda) \triangleq \mathcal{E}(\hat{w}(\lambda), \lambda) \right\} \qquad \text{s.c.} \qquad \hat{w}(\lambda) = \operatorname*{argmin}_{w} \mathcal{L}(w; \lambda)$$

On peut voir le problème supérieur comme visant à minimiser un objectif $\mathcal{U}$

$$\begin{array}{l} \text{Choix initial } \lambda_0 \in \Lambda \\ \text{Pour } \; q = 0, \dots, \\ \lfloor \; \lambda_{q+1} = \operatorname{Proj}_{\Lambda}\left(\lambda_q - \eta \nabla \mathcal{U}(\lambda_q)\right) \end{array}$$

Problèmes potentiels

- Cela suppose que l'on connaisse $\hat{w}(\lambda)$ de manière exacte !
- $\mathcal{U}$ n'est pas forcément différentiable
- Comment choisir le pas η ? Est-ce que $\mathcal{U}$ est localement lisse ?



$$w^{(k)}(\lambda) \underset{k\to\infty}{\longrightarrow} \hat{w}(\lambda) = \underset{w}{\operatorname{argmin}}\, \mathcal{L}(w, \lambda)$$

Problème bi-niveaux approché

$$\min_{\lambda\in\Lambda} \mathcal{E}(w^{(k)}(\lambda), \lambda) \qquad \text{s.c.} \qquad w^{(k)}(\lambda) \underset{k\to\infty}{\longrightarrow} \hat{w}(\lambda) = \underset{w}{\operatorname{argmin}}\, \mathcal{L}(w; \lambda)$$

Qualité de l'approximation

Problème exact

$$\min_{\lambda\in\Lambda} \left\{\mathcal{U}(\lambda) \triangleq \mathcal{E}(\hat{w}(\lambda),\lambda)\right\}$$

$$\text{s.c.} \quad \hat{w}(\lambda) = \operatorname*{argmin}_{w\in\mathcal{W}} \mathcal{L}(w;\lambda)$$

Problème approché

$$\min_{\lambda\in\Lambda} \left\{\mathcal{U}^{(k)}(\lambda) \triangleq \mathcal{E}(w^{(k)}(\lambda),\lambda)\right\}$$

$$\text{s.c.} \quad \begin{array}{l} \text{for } i = 0,1,\ldots,k-1 \\ \lfloor \; w^{(i+1)}(\lambda) = \mathcal{A}(w^{(i)}(\lambda),\lambda) \\ w^{(k)}(\lambda) \to \hat{w}(\lambda) \end{array}$$

Problème exact

$$\min_{\lambda\in\Lambda} \left\{ \mathcal{U}(\lambda) \triangleq \mathcal{E}(\hat{w}(\lambda), \lambda) \right\}$$

$$\text{s.c.} \quad \hat{w}(\lambda) = \operatorname*{argmin}_{w\in\mathcal{W}} \mathcal{L}(w;\lambda)$$

Problème approché

$$\min_{\lambda\in\Lambda} \left\{ \mathcal{U}^{(k)}(\lambda) \triangleq \mathcal{E}(w^{(k)}(\lambda), \lambda) \right\}$$

$$\text{s.c.} \quad \begin{array}{l} \text{for } i = 0, 1, \ldots, k-1 \\ \lfloor \; w^{(i+1)}(\lambda) = \mathcal{A}(w^{(i)}(\lambda), \lambda) \\ w^{(k)}(\lambda) \to \hat{w}(\lambda) \end{array}$$

Théorème informel

Si $w^{(k)}(\lambda) \underset{k\to\infty}{\longrightarrow} \hat{w}(\lambda)$ uniformément, alors les minima et la valeur infimum de $\mathcal{U}^{(k)}$ convergent vers ceux de $\mathcal{U}$

« J. Frecon, S. Salzo, and M. Pontil. Bilevel learning of the group lasso structure. In Advances in Neural Information Processing Systems 31 (NeurIPS), pages 8301–8311. 2018 »

Problème exact

$$\min_{\lambda \in \Lambda} \left\{ \mathcal{U}(\lambda) \triangleq \mathcal{E}(\hat{w}(\lambda), \lambda) \right\}$$

$$\text{s.c.} \quad \hat{w}(\lambda) = \operatorname*{argmin}_{w \in \mathcal{W}} \mathcal{L}(w; \lambda)$$

Problème approché

$$\min_{\lambda \in \Lambda} \left\{ \mathcal{U}^{(k)}(\lambda) \triangleq \mathcal{E}(w^{(k)}(\lambda), \lambda) \right\}$$

$$\text{s.c.} \quad \begin{array}{l} \text{for } i = 0, 1, \ldots, k-1 \\ \lfloor \; w^{(i+1)}(\lambda) = \mathcal{A}(w^{(i)}(\lambda), \lambda) \\ w^{(k)}(\lambda) \to \hat{w}(\lambda) \end{array}$$

Les propriétés de $\mathcal{U}$ dépendent de l'objectif $\mathcal{L}$

Les propriétés de $\mathcal{U}^{(k)}$ dépendent de l'algorithme $\mathcal{A}$

Certaines propriétés de $\mathcal{A}$ dépendent de $\mathcal{L}$

Dans la suite, nous allons considérer les problèmes bi-niveaux exacts et approchés selon les cas où ils sont différentiables ou non

Optimisation bi-niveaux basée sur le gradient

- Cas différentiable -

Problème bi-niveaux exact et différentiable

$$\min_{\lambda\in\Lambda} \left\{\mathcal{U}(\lambda) \triangleq \mathcal{E}(\hat{w}(\lambda),\lambda)\right\} \qquad \text{s.c.} \qquad \hat{w}(\lambda) = \operatorname*{argmin}_{w} \mathcal{L}(w;\lambda)$$

Différentiation implicite

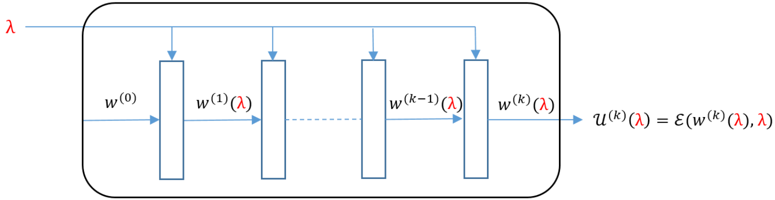
« Almeida, L. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In Proceedings, 1st First International Conference on Neural Networks 1987 »

« Pineda, F. J. Generalization of back-propagation to recurrent neural networks . In Physical Review Letters 1987 »

« Domke, J. Generic methods for optimization-based modeling. In AISTATS 2012 »

« Pedregosa, F. Hyperparameter optimization with approximate gradient. In ICML 2016 »

Problème bi-niveaux approché et différentiable



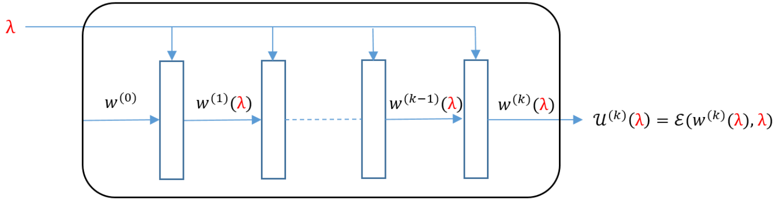
$$w^{(k)}(\lambda) \underset{k\to\infty}{\longrightarrow} \hat{w}(\lambda) = \underset{w}{\operatorname{argmin}}\ \mathcal{L}(w, \lambda)$$

$\mathcal{L}$ différentiable $\Rightarrow$ Algorithme généralement différentiable

$\Rightarrow$ Rétropropagation du gradient

$$\nabla\mathcal{U}^{(k)}(\lambda) =? \longrightarrow \text{Dérivation en chaîne de } w^{(i+1)}(\lambda) = \mathcal{A}(w^{(i)}(\lambda), (\lambda))$$

Problème bi-niveaux approché et différentiable



$$w^{(k)}(\lambda) \underset{k\to\infty}{\longrightarrow} \hat{w}(\lambda) = \underset{w}{\operatorname{argmin}}\ \mathcal{L}(w, \lambda)$$

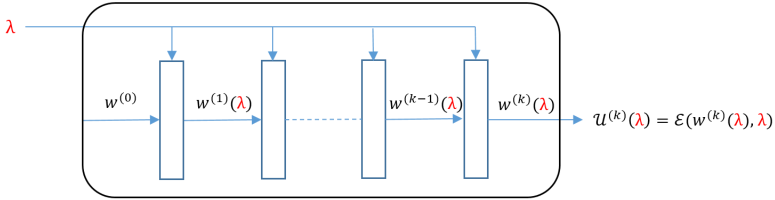
Différentiation déroulée

« Griewank, A. W. Evaluating Derivatives : Principles and Techniques of Algorithmic Differentiation. In Cambridge press 2008 »

« Maclaurin, D., Duvenaud, D., and Adams, R. P. Gradient-based hyperparameter optimization through reversible learning. In ICML 2015 »

« Franceschi, L., Donini, M., Frasconi, P., and Pontil, M. Forward and reverse gradient-based hyper- parameter optimization. In ICML 2017 »

Problème bi-niveaux approché et différentiable

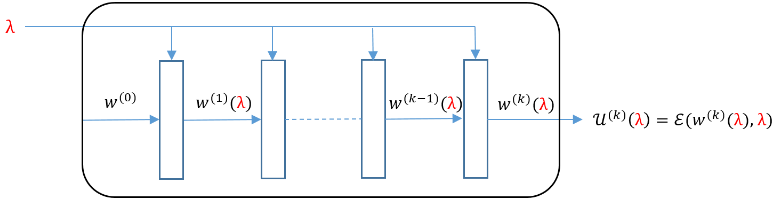


$$w^{(k)}(\lambda) \underset{k\to\infty}{\longrightarrow} \hat{w}(\lambda) = \underset{w}{\operatorname{argmin}}\ \mathcal{L}(w, \lambda)$$

Différentiation déroulée tronquée

« Shaban, A., Cheng, C.-A., Hatch, N., and Boots, B. Truncated back-propagation for bilevel optimiza- tion. In AISTATS 2019 »

Problème bi-niveaux approché et différentiable



$$w^{(k)}(\lambda) \underset{k\to\infty}{\longrightarrow} \hat{w}(\lambda) = \underset{w}{\operatorname{argmin}}\ \mathcal{L}(w, \lambda)$$

Approximation à la dernière itération

« Lorraine, J., Vicol, P., and Duvenaud, D. Optimizing millions of hyperparameters by implicit differentiation. In AISTATS 2020 »

Optimisation bi-niveaux basée sur le gradient

- Cas non différentiable -

Problème bi-niveaux exact et non différentiable

Problème bi-niveaux exact et non différentiable

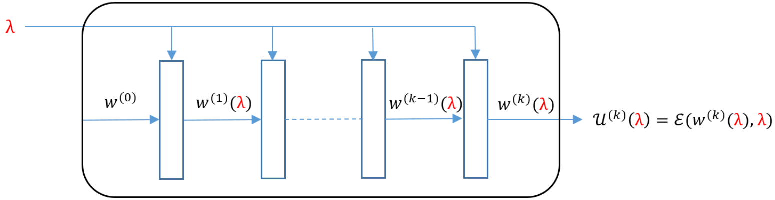
$$\min_{\lambda \in \Lambda} \left\{ \mathcal{U}(\lambda) \triangleq \mathcal{E}(\hat{w}(\lambda), \lambda) \right\} \qquad \text{s.c.} \qquad \hat{w}(\lambda) = \operatorname*{argmin}_{w} \mathcal{L}(w; \lambda)$$

Analyse variationnelle de l'application $\lambda \mapsto \hat{w}(\lambda)$
Dérivation des conditions d'optimalité

« B. Mordukhovich. Variational analysis and applications. In Springer Monographs in Mathematics, 2018. »

« J.C. De los Reyes, D. Villacís. Optimality Conditions for Bilevel Imaging Learning Problems with Total Variation Regularization. arXiv preprint. 2021. »

Problème bi-niveaux approché et non-différentiable



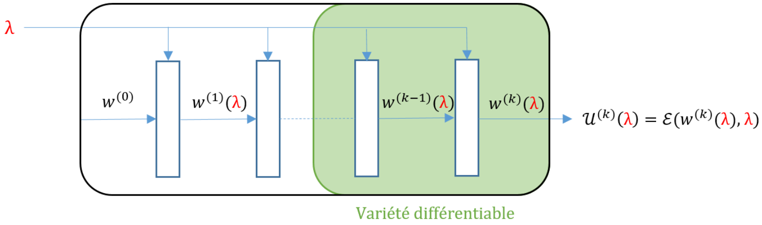
$$w^{(k)}(\lambda) \underset{k\to\infty}{\longrightarrow} \hat{w}(\lambda) = \operatorname*{argmin}_{w} \mathcal{L}(w, \lambda)$$

$\mathcal{L}$ non-différentiable $\Rightarrow$ Algorithme généralement non-différentiable

$\Rightarrow$ Rétropropagation du gradient instable

« S. M. Kakade, and J. D. Lee Provably correct automatic subdifferentiation for qualified programs. In NIPS 2018 »

Problème bi-niveaux approché et non-différentiable



$$w^{(k)}(\lambda) \underset{k\to\infty}{\longrightarrow} \hat{w}(\lambda) = \underset{w}{\operatorname{argmin}}\ \mathcal{L}(w, \lambda)$$

Différentiation dans les variétés différentiables

« Q. Bertrand, Q. Klopfenstein, M. Blonde, S. Vaiter, A. Gramfort, and J. Salmon Implicit differentiation of Lasso-type models for hyperparameter optimization. In ICML 2020 »

$$\hat{w}(\lambda) = \underset{w}{\operatorname{argmin}} \underbrace{\frac{1}{2}\|y - Xw\|^2}_{f(w)} + \underbrace{\lambda\|w\|_1}_{g(w)}$$

Résolution par minimisation successive de majorants quadratiques

Choix initial $w^{(0)}$

Pour $k = 0, \ldots$

$\lfloor\; w^{(k+1)} = \operatorname{argmin}_w f(w^{(k)}) + \nabla f(w^{(k)})^\top (w - w^{(k)}) + \frac{1}{2\tau}\|w - w^{(k)}\|^2 + g(w)$

$$\hat{w}(\lambda) = \underset{w}{\operatorname{argmin}} \underbrace{\frac{1}{2}\|y - Xw\|^2}_{f(w)} + \underbrace{\lambda\|w\|_1}_{g(w)}$$

Se réduit à un algorithme forward-backward

Choix initial $w^{(0)}$

Pour $k = 0, \ldots$

$\lfloor\ w^{(k+1)} = \operatorname{prox}_{\tau g}(w^{(k)} - \tau \nabla f(w^{(k)}))$

Aparté : exemple de variété différentiable

$$\hat{w}(\lambda) = \underset{w}{\operatorname{argmin}} \underbrace{\frac{1}{2}\|y - Xw\|^2}_{f(w)} + \underbrace{\lambda\|w\|_1}_{g(w)}$$

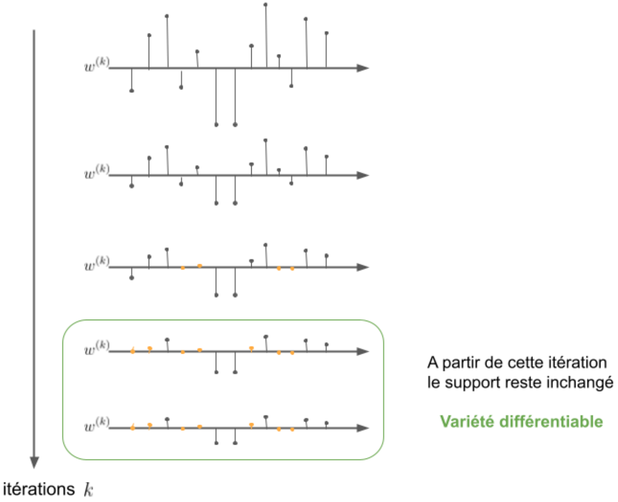
Le tout est en fait l'algorithme de descente de gradient seuillé

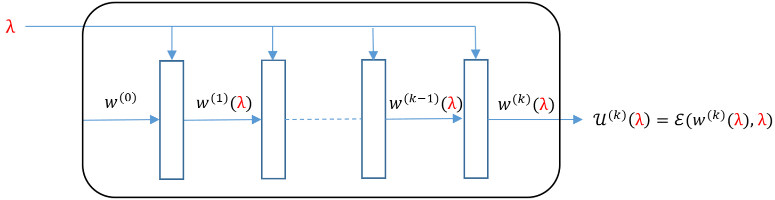
Choix initial $w^{(0)}$

Pour $k = 0, \ldots$

$\lfloor \; w^{(k+1)} = \mathrm{Soft}_{\tau\lambda}(w^{(k)} - \tau\nabla f(w^{(k)}))$

En appliquant cet algorithme, on observe ...





$$w^{(k)}(\lambda) \underset{k\to\infty}{\longrightarrow} \hat{w}(\lambda) = \underset{w}{\operatorname{argmin}}\ \mathcal{L}(w, \lambda)$$

Algorithme différentiables pour résoudre des problèmes non-différentiables

« J. Frecon, S. Salzo, and M. Pontil. Bilevel learning of the group lasso structure. In Advances in Neural Information Processing Systems 31 (NeurIPS), pages 8301–8311. 2018 »

« J. Frecon, S. Salzo, and M. Pontil. Unveiling groups of related tasks in multi-task learning. In Proceedings of the International Conference on Pattern Recognition (ICPR). 2020 »

« J. Frecon, S. Salzo, and M. Pontil. Bilevel optimization of groupwise penalties. En préparation. 2022 »

Aparte : comment obtenir un algorithme differentiable

$$\hat{w} = \underset{w}{\operatorname{argmin}} f(w) + g(w)$$

Résolution par minimisation successive de majorants *bien choisis*

Choix initial $w^{(0)}$
Pour $k = 0, \ldots$
$\lfloor \; w^{(k+1)} = \operatorname{argmin}_w f(w^{(k)}) + \nabla f(w^{(k)})^\top (w - w^{(k)}) + \frac{1}{\tau} D(w, w^{(k)}) + g(w)$

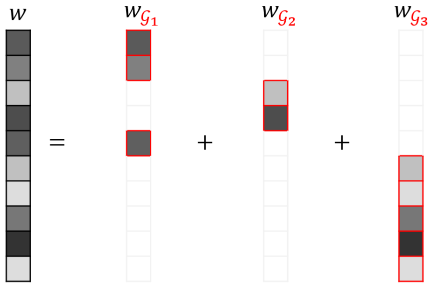
Le choix de la distance $D(w, w^{(k)})$ peut permettre d'obtenir un algorithme différentiable

Application à la découverte de structure

Application à la découverte de structure

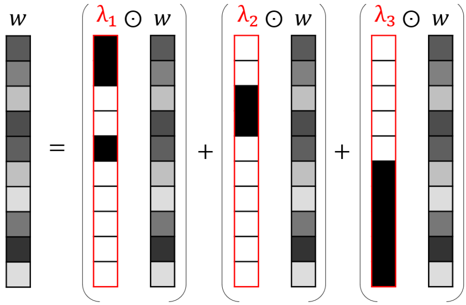
La solution du *Group Lasso* [Yuan and Lin (2006)] dépend des groupes $\{\mathcal{G}_1, \ldots, \mathcal{G}_L\}$

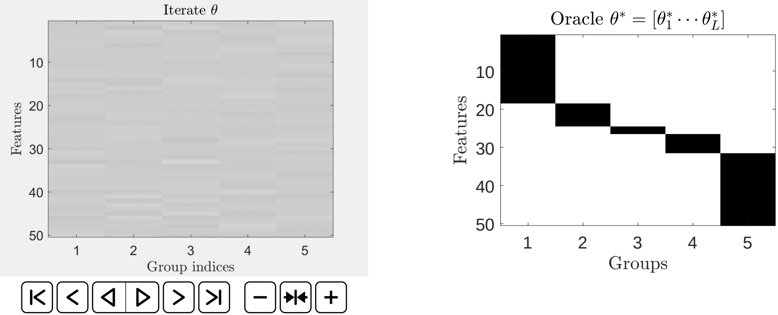
$$\hat{w}(\mathcal{G}_1, \ldots, \mathcal{G}_L) = \underset{w \in \mathcal{W}}{\operatorname{argmin}} \; \ell_{\mathrm{trn}}(w) + \gamma \sum_{l=1}^{L} \|w_{\mathcal{G}_l}\|_2$$



Formalisme bi-niveaux pour apprendre l'indicatrice $\lambda = [\lambda_1 \cdots \lambda_L]$ des groupes

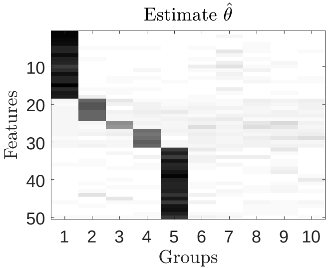
$$\min_{\lambda=[\lambda_1\cdots\lambda_L]\in\Lambda} \ell_{\mathrm{val}}(\hat{w}(\lambda)) \quad \text{s.c.} \quad \hat{w}(\lambda) = \operatorname*{argmin}_{w\in\mathcal{W}} \ \ell_{\mathrm{trn}}(w) + \gamma \sum_{l=1}^{L} \|\lambda_l \odot w\|_2$$

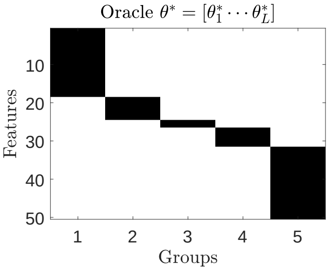




On retrouve les bons groupes ! (modulo une permutation)

Quand le nombre de groupes est inconnu





Identifie correctement les groupes inutiles

Régularisation structurée : $\lambda =$ groupes de variables

« J. Frecon, S. Salzo, and M. Pontil. Bilevel optimization of groupwise penalties. En préparation. 2022 »

$$\min_{\color{red}{\lambda=[\lambda_1\cdots\lambda_L]\in\Lambda}} \ell_{\mathrm{val}}(\hat{w}(\color{red}{\lambda})) \quad \text{s.c.} \quad \hat{w}(\color{red}{\lambda}) = \operatorname*{argmin}_{w\in\mathcal{W}} \ell_{\mathrm{trn}}(w) + \underbrace{\sum_{l=1}^{L}\Omega_l(w,\color{red}{\lambda_l})}_{\text{apprendre la structure de la régularisation}}$$

Théorème informel

Les itérées convergent vers un point stationnaire de l'erreur de validation

Contributions en apprentissage bi-niveaux

Régularisation structurée : $\lambda =$ groupes de variables

« J. Frecon, S. Salzo, and M. Pontil. Bilevel optimization of groupwise penalties. En préparation. 2021 »

Apprentissage multi-tâches : $\lambda =$ groupes de tâches

« J. Frecon, S. Salzo, and M. Pontil. Unveiling groups of related tasks in multi-task learning. In Proceedings of the International Conference on Pattern Recognition (ICPR). 2020 »

Tâche 1

Tâche 2

Tâche 3

Tâche 4

Tâche 5

Régularisation structurée : $\lambda =$ groupes de variables

« J. Frecon, S. Salzo, and M. Pontil. Bilevel optimization of groupwise penalties. En préparation. 2021 »

Apprentissage multi-tâches : $\lambda =$ groupes de tâches

« J. Frecon, S. Salzo, and M. Pontil. Unveiling groups of related tasks in multi-task learning. In Proceedings of the International Conference on Pattern Recognition (ICPR). 2020 »

Contributions en apprentissage bi-niveaux

Régularisation structurée : $\lambda =$ groupes de variables

« J. Frecon, S. Salzo, and M. Pontil. Bilevel optimization of groupwise penalties. En préparation. 2021 »

Apprentissage multi-tâches : $\lambda =$ groupes de tâches

« J. Frecon, S. Salzo, and M. Pontil. Unveiling groups of related tasks in multi-task learning. In Proceedings of the International Conference on Pattern Recognition (ICPR). 2020 »

Tâche 1

Tâche 2

Tâche 3

Tâche 4

Tâche 5

Théorème informel

L'opérateur proximal de la norme spectrale est différentiable pour une classe de distances de Bregman

Contributions en apprentissage bi-niveaux

Régularisation structurée : $\lambda =$ groupes de variables

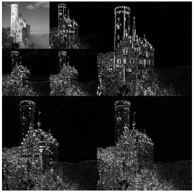
« J. Frecon, S. Salzo, and M. Pontil. Bilevel optimization of groupwise penalties. En préparation. 2021 »

Apprentissage multi-tâches : $\lambda =$ groupes de tâches

« J. Frecon, S. Salzo, and M. Pontil. Unveiling groups of related tasks in multi-task learning. In Proceedings of the International Conference on Pattern Recognition (ICPR). 2020 »

Apprentissage d'ondelette : $\lambda =$ filtre d'ondelette

« J. Frecon, R. Grazzi, S. Salzo, and M. Pontil. Smooth optimization of wavelet bases. Submitted to Conférence sur l'Apprentissage Automatique (CAp). 2021 »



double objectif :

- Réduire le nombre de coefficients d'ondelettes
- Diminuer l'erreur de reconstruction

Conclusion et perspectives

Problème exact

$$\min_{\lambda\in\Lambda} \left\{\mathcal{U}(\lambda) \triangleq \mathcal{E}(\hat{w}(\lambda), \lambda)\right\}$$

$$\text{s.c.} \quad \hat{w}(\lambda) = \operatorname*{argmin}_{w\in\mathcal{W}} \mathcal{L}(w;\lambda)$$

Problème approché

$$\min_{\lambda\in\Lambda} \left\{\mathcal{U}^{(k)}(\lambda) \triangleq \mathcal{E}(w^{(k)}(\lambda), \lambda)\right\}$$

$$\text{s.c.} \quad \begin{array}{l} \text{for } i = 0, 1, \ldots, k-1 \\ \lfloor \; w^{(i+1)}(\lambda) = \mathcal{A}(w^{(i)}(\lambda), \lambda) \\ w^{(k)} \to \hat{w}(\lambda) \end{array}$$

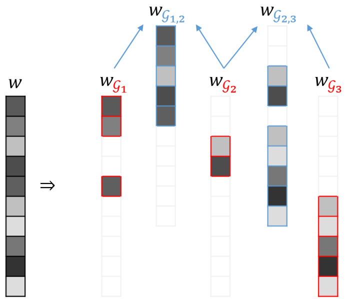
$$\lambda^{(n+1)} = \mathrm{Proj}_{\Lambda}(\lambda^{(n)} - \mu\nabla\mathcal{U}^{(k)}(\lambda^{(n)}))$$

- Convergence de $\{\mathcal{U}^{(k)}\}_k$ vers $\mathcal{U}$? (inf ✓, argmin ✓, points stationnaires, ...)
- Convergence de $\{\nabla\mathcal{U}^{(k)}(\lambda)\}_k$ vers un sous-gradient de $\partial\mathcal{U}(\lambda)$?
- Convergence de $\{\lambda^{(n)}\}_n$ vers un point critique de $\mathcal{U}$?
- Optimisation stochastique
- Calcul efficace d'hypergradient $\nabla\mathcal{U}^{(k)}(\lambda^{(n)})$

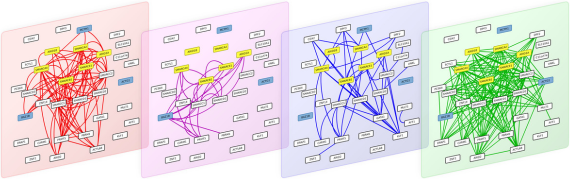
Extension du formalisme bi-niveaux
$\Rightarrow$ apprendre des structures de graphe $\{\mathcal{G}_l, \mathcal{G}_{l,l'}\}$ sur les paramètres à estimer

$$\min_{\{\mathcal{G}_l, \mathcal{G}_{l,l'}\}} \mathcal{E}\Big(\hat{w}(\{\mathcal{G}_l, \mathcal{G}_{l,l'}\})\Big)$$

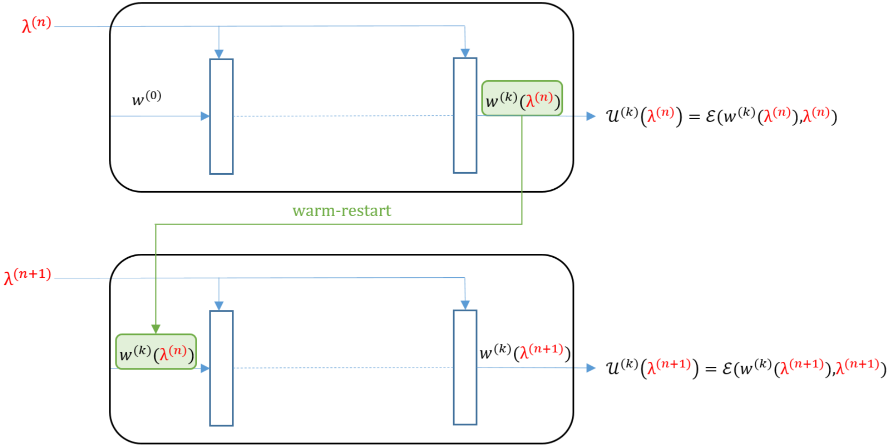
$$\text{s.c.} \quad \hat{w}(\{\mathcal{G}_l, \mathcal{G}_{l,l'}\}) = \operatorname*{argmin}_{w \in \mathcal{W}} \ell_{\mathrm{trn}}(w) + \sum_{l=1}^{L} \gamma_l \Omega_l(w_{\mathcal{G}_l}) + \sum_{l \neq l'} \gamma_{l,l'} \Omega_{l,l'}(w_{\mathcal{G}_{l,l'}})$$



Application : identification de communautés dans les réseaux biologiques



Communautés de protéines associées au syndrome de Coffin-Siris [Didier et al., 2015]


$\lambda^{(n)}$
$w^{(0)}$
$w^{(k)}(\lambda^{(n)})$
$\mathcal{U}^{(k)}(\lambda^{(n)}) = \mathcal{E}(w^{(k)}(\lambda^{(n)}),\lambda^{(n)})$
warm-restart
$\lambda^{(n+1)}$
$w^{(k)}(\lambda^{(n)})$
$w^{(k)}(\lambda^{(n+1)})$
$\mathcal{U}^{(k)}(\lambda^{(n+1)}) = \mathcal{E}(w^{(k)}(\lambda^{(n+1)}),\lambda^{(n+1)})$

$$\min_{\lambda \in \Lambda} \mathop{\mathbb{E}}_{x \sim val} \left[\mathcal{E}\left(\hat{w}(\lambda), \lambda; x\right) \right] \qquad \text{s.c.} \qquad \hat{w}(\lambda) \in \operatorname*{argmin}_{w \in \mathcal{W}} \mathop{\mathbb{E}}_{x \sim trn} \left[\mathcal{L}\left(w, \lambda; x\right) \right]$$

Garanties numériques pour l'optimisation stochastique

Extension à la validation croisée (k-fold, Monte-Carlo)

Développement de bornes statistiques

Merci pour votre attention

Formalisme bi-niveaux

Problème exact

$$\min_{\lambda\in\Lambda} \left\{\mathcal{U}(\lambda) \triangleq \mathcal{E}(\hat{w}(\lambda),\lambda)\right\}$$

$$\text{s.c.} \quad \hat{w}(\lambda) = \operatorname*{argmin}_{w\in\mathcal{W}} \mathcal{L}(w;\lambda)$$

$\hat{w}(\lambda)$ sans expression explicite

Problème approché

$$\min_{\lambda\in\Lambda} \left\{\mathcal{U}^{(k)}(\lambda) \triangleq \mathcal{E}(w^{(k)}(\lambda),\lambda)\right\}$$

s.c.
- $w^{(0)}(\lambda)$ choisi arbitrairement
- for $i = 0, 1, \ldots, k-1$
  - $w^{(i+1)}(\lambda) = \mathcal{A}(w^{(i)}(\lambda),\lambda)$
- $w^{(k)}(\lambda) \to \hat{w}(\lambda)$

$\mathcal{U}^{(k)}$ lisse si $\mathcal{A}$ lisse

$$\lambda^{(n+1)} = \mathrm{Proj}_{\Lambda}(\lambda^{(n)} - \mu\nabla\mathcal{U}^{(k)}(\lambda^{(n)}))$$

- $\inf_{\lambda\in\Lambda} \mathcal{U}^{(k)}(\lambda) \xrightarrow[k\to+\infty]{} \inf_{\lambda\in\Lambda} \mathcal{U}(\lambda)$?
- $\operatorname*{argmin}_{\lambda\in\Lambda} \mathcal{U}^{(k)}(\lambda) \xrightarrow[k\to+\infty]{} \operatorname*{argmin}_{\lambda\in\Lambda} \mathcal{U}(\lambda)$?
- $\lim_{k\to\infty} \nabla\mathcal{U}^{(k)}(\lambda) \in \partial\mathcal{U}(\lambda)$?
- Impact de "warm-restart" sur $w^{(0)}$?
- Calcul efficace de $\nabla\mathcal{U}^{(k)}$?
- $\lim_{n\to\infty} \lambda^{(n)} \in \partial\mathcal{U}^{-1}(0)$?

Etude des problèmes bi-niveaux non différentiables

$$\min_{\lambda \in \Lambda} \mathop{\mathbb{E}}_{x \sim val} \left[\mathcal{E}\left(\hat{w}(\lambda), \lambda; x\right) \right] \qquad \text{s.c.} \qquad \hat{w}(\lambda) \in \operatorname*{argmin}_{w \in \mathcal{W}} \mathop{\mathbb{E}}_{x \sim trn} \left[\mathcal{L}\left(w, \lambda; x\right) \right]$$

Développement de bornes statistiques

Garanties numériques pour l'optimisation stochastique

Extension à la validation croisée (k-fold, Monte-Carlo)

Setting : T linear regression tasks arranged in L groups of related tasks $\{\mathcal{G}_1, \ldots, \mathcal{G}_L\}$

find w_1	find w_2	find w_3	find w_4	find w_5
$y_1 \approx X_1 w_1$	$y_2 \approx X_2 w_2$	$y_3 \approx X_3 w_3$	$y_4 \approx X_4 w_4$	$y_5 \approx X_5 w_5$

$W_{\mathcal{G}_1} = [w_1 \, w_2]$
low-rank

$W_{\mathcal{G}_2} = [w_3 \, w_4 \, w_5]$
low-rank

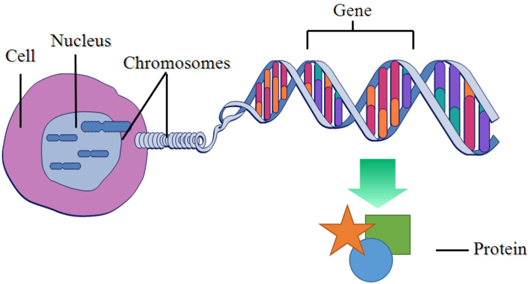
$$\hat{W} \in \underset{W=[w_1 \cdots w_T]}{\operatorname{argmin}} \sum_{t=1}^{T} \frac{1}{2} \|y_t - X_t w_t\|^2 + \lambda \sum_{l=1}^{L} \|W_{\mathcal{G}_l}\|_{\mathrm{tr}}$$

Issue : In practice we don't know how tasks are related → need to estimate $\{\mathcal{G}_1, \ldots, \mathcal{G}_L\}$

Motivation : genes expression analysis

Goal : Predict the function of proteins from regulatory patterns

Collaboration with Giorgio Valentini (Universita degli Studi di Milano)



Gene = long sequence with regulatory patterns (dictate gene expression)

Proteins perform various functions (transport, redox, binding ...)

Motivation : genes expression analysis

Regulatory patterns

	R_1	R_2	R_3	...	R_P
	CTGAC	GGATC	GCAAG	...	ATCAG
Gene 1	1	1	0	...	0

neuron

Protein sub-functions (Gene Ontology)

Go_1	Go_2	Go_3	...	Go_T
transport	redox	binding		
1	0	0	...	1

Sequences R_1 and R_2 are present in Gene 1

Gene 1 produces neurons

Neurons perform transport of electrons

Regulatory patterns

	R_1 CTGAC	R_2 GGATC	R_3 GCAAG	... ...	R_P ATCAG
Gene 1	1	1	0	...	0
Gene 2	1	1	1	...	0

neuron =

hormone =

Protein sub-functions (Gene Ontology)

Go_1 transport	Go_2 redox	Go_3 binding	...	Go_T
1	0	0	...	1
1	1	0	...	0

Sequences R_1, R_2 and R_3 are present in Gene 2

Gene 2 produces hormones

Hormones perform transport of particles and reduction–oxidation

Motivation : genes expression analysis

	Regulatory patterns						Protein sub-functions (Gene Ontology)				
	R_1 CTGAC	R_2 GGATC	R_3 GCAAG	... ...	R_P ATCAG		Go_1 transport	Go_2 redox	Go_3 binding	...	Go_T
Gene 1	1	1	0	...	0	neuron	1	0	0	...	1
Gene 2	1	1	1	...	0	hormone	1	1	0	...	0
Gene 3	0	0	0	...	1	antibody	0	0	1	...	0

Sequence R_P is present in Gene 3

Gene 3 produces antibodies

Antibodies perform binding of particles

Motivation : genes expression analysis

Regulatory patterns

	R_1 CTGAC	R_2 GGATC	R_3 GCAAG	... ...	R_P ATCAG
Gene 1	1	1	0	...	0
Gene 2	1	1	1	...	0
Gene 3	0	0	0	...	1
	⋮	⋮	⋮	⋮	⋮
Gene N	1	1	0	...	1

$X \in \{0,1\}^{N \times P}$

neuron

hormone

antibody

Protein sub-functions (Gene Ontology)

Go_1 transport	Go_2 redox	Go_3 binding	...	Go_T
1	0	0	...	1
1	1	0	...	0
0	0	1	...	0
⋮	⋮	⋮	⋮	⋮
0	0	1	...	1

$y = [y_1 \cdots y_T] \in \{0,1\}^{N \times T}$

Regulatory patterns

	R_1 CTGAC	R_2 GGATC	R_3 GCAAG	... ...	R_P ATCAG
Gene 1	1	1	0	...	0
Gene 2	1	1	1	...	0
Gene 3	0	0	0	...	1
	⋮	⋮	⋮	⋮	⋮
Gene N	1	1	0	...	1

$X \in \{0,1\}^{N \times P}$

neuron =

hormone =

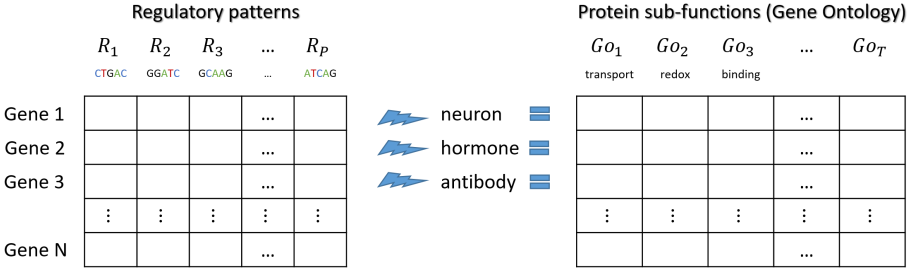
antibody =

Protein sub-functions (Gene Ontology)

Go_1 transport	Go_2 redox	Go_3 binding	...	Go_T
1	0	0	...	1
1	1	0	...	0
0	0	1	...	0
⋮	⋮	⋮	⋮	⋮
0	0	1	...	1

$y = [y_1 \cdots y_T] \in \{0,1\}^{N \times T}$

Goal 1 : Predict each y_t from X



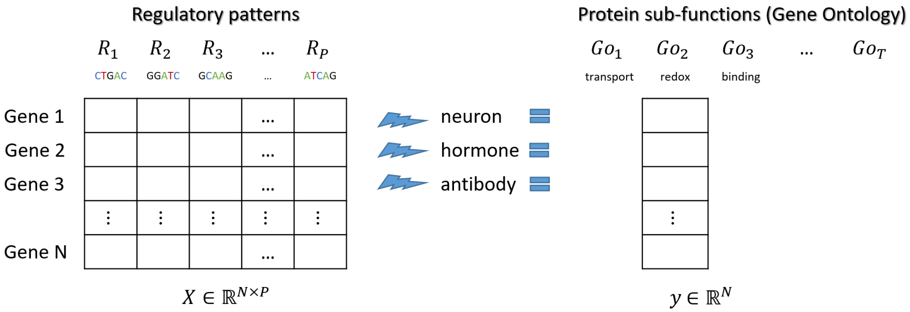
$X \in \mathbb{R}^{N \times P}$

$y = [y_1 \cdots y_T] \in \mathbb{R}^{N \times T}$

Goal 1 : Predict each y_t from X

$\rightarrow$ to generalize : X and y are not only made of 0's and 1's

Motivation : genes expression analysis

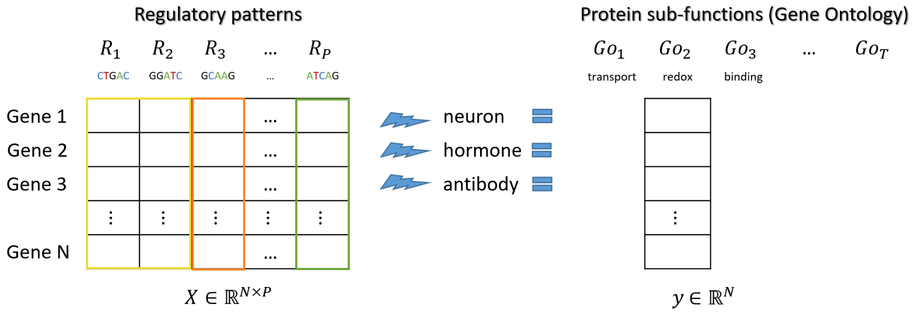


Goal 1 : Predict y from X

$\rightarrow$ to generalize : X and y are not only made of 0's and 1's

$\rightarrow$ to simplify : we first assume that $T = 1$ and omit the index t

Motivation : genes expression analysis



Goal 1 : Predict y from X

$\rightarrow$ to generalize : X and y are not only made of 0's and 1's
$\rightarrow$ to simplify : we first assume that $T = 1$ and omit the index t

Goal 2 : Discover if there exist some groups in X
ex : R_1 and R_2 are both equally relevant to predict y

Assumptions

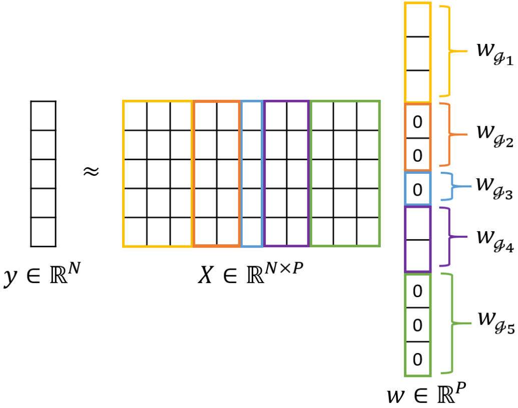
Model the observations

$\implies$ linear model + Gaussian distribution : there exists w such that $y \sim \mathcal{N}(Xw, \sigma)$

Model the group structure

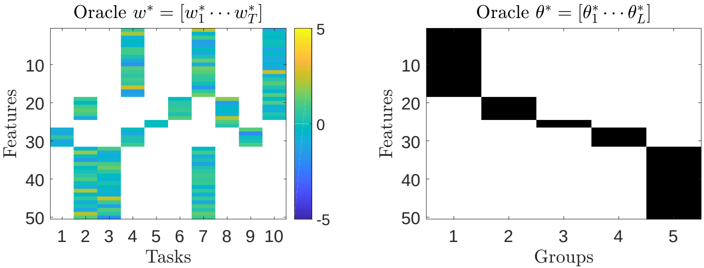
few groups of features in X are relevant to predict y

$\implies$ group sparsity : some groups of variables in w are zero while others are non-zero



Configuration : T problèmes Group Lasso partageant la même structure de groupe

$$(\forall t \in \{1,\ldots,T\}) \quad \hat{w}_t(\theta) \in \underset{w_t \in \mathbb{R}^P}{\operatorname{argmin}} \frac{1}{2}\|y_t - X_t w_t\|^2 + \lambda \sum_{l=1}^{L} \|w_t \odot \theta_l\|_2,$$



But : Estimer la structure de groupe optimale θ^*

Problème primal

$$\underset{w\in\mathbb{R}^P}{\text{minimize}}\left\{\ell(w,\theta) := \underbrace{\frac{1}{2}\|y - Xw\|_2^2 + \frac{\epsilon}{2}\|w\|_2^2}_{f(w)\ \text{différentiable}} + \underbrace{\lambda\sum_{l=1}^{L}\|\theta_l \odot w\|_2}_{g(A_\theta w)\ \text{non différentiable}}\right\}$$

où $A_\theta : w \in \mathbb{R}^P \mapsto (\theta_1 \odot w, \ldots, \theta_L \odot w) \in \mathbb{R}^{P\times L}$

Algorithme forward-backward

$$\left\{\begin{array}{l} \text{for } q = 0, 1, \ldots, Q-1 \\ \quad \left\lfloor\, w^{(q+1)}(\theta) = \mathrm{prox}_{\beta g\circ A_\theta}\left(w^{(q)}(\theta) - \beta\nabla f(w^{(q)}(\theta))\right)\right. \end{array}\right.$$

Difficulté : $\mathrm{prox}_{g\circ A_\theta}(v) = \underset{w\in\mathbb{R}^P}{\mathrm{argmin}}\ g(A_\theta w) + \frac{1}{2}\|w - v\|_2^2$ sans expression explicite

Problème primal

$$\operatorname*{minimize}_{w\in\mathbb{R}^P}\left\{\ell(w,\theta) := \underbrace{\frac{1}{2}\|y - Xw\|_2^2 + \frac{\epsilon}{2}\|w\|_2^2}_{f(w)\text{ différentiable}} + \underbrace{\lambda\sum_{l=1}^{L}\|\theta_l \odot w\|_2}_{g(A_\theta w)\text{ non différentiable}}\right\}$$

où $A_\theta : w \in \mathbb{R}^P \mapsto (\theta_1 \odot w, \dots, \theta_L \odot w) \in \mathbb{R}^{P\times L}$

Algorithme forward-backward

$$\left\{\begin{array}{l}\text{for } q = 0, 1, \dots, Q-1\\ \quad \left\lfloor\; w^{(q+1)}(\theta) = \operatorname{prox}_{\beta g\circ A_\theta}\left(w^{(q)}(\theta) - \beta\nabla f(w^{(q)}(\theta))\right)\right.\end{array}\right.$$

Difficulté : $\operatorname{prox}_{g\circ A_\theta}(v) = \operatorname*{argmin}_{w\in\mathbb{R}^P}\, g(A_\theta w) + \frac{1}{2}\|w - v\|_2^2$ sans expression explicite

Problème primal

$$\underset{w\in\mathbb{R}^P}{\text{minimize}}\left\{\ell(w,\theta) := \underbrace{\frac{1}{2}\|y - Xw\|_2^2 + \frac{\epsilon}{2}\|w\|_2^2}_{f(w)} + \underbrace{\lambda\sum_{l=1}^{L}\|\theta_l \odot w\|_2}_{g(A_\theta w)}\right\}$$

$\boxed{\mathrm{prox}_{g\circ A_\theta} \text{ n'a pas de forme explicite} \Rightarrow \text{considère le problème dual pour déplacer } A_\theta}$

Problème dual

$$\underset{u\in\mathbb{R}^{P\times L}}{\text{minimize}}\left\{\tilde{\ell}(u,\theta) := f^*(-A_\theta^* u) + g^*(u)\right\}$$

où

A_θ^* est l'adjoint de A_θ

f^* et g^* sont les conjuguées de Fenchel de f et g, respectivement.

Algorithme lisse

Problème primal

$$\underset{w\in\mathbb{R}^P}{\text{minimize}} \left\{ \ell(w,\theta) := \underbrace{\frac{1}{2}\|y - Xw\|_2^2 + \frac{\epsilon}{2}\|w\|_2^2}_{f(w)} + \underbrace{\lambda \sum_{l=1}^{L} \|\theta_l \odot w\|_2}_{g(A_\theta w)} \right\}$$

$\boxed{\text{prox}_{g\circ A_\theta} \text{ n'a pas de forme explicite} \Rightarrow \text{considère le problème dual pour déplacer } A_\theta}$

Problème dual

$$\underset{u\in\mathbb{R}^{P\times L}}{\text{minimize}} \left\{ \tilde{\ell}(u,\theta) := f^*(-A_\theta^* u) + g^*(u) \right\}$$

où

A_θ^* est l'adjoint de A_θ

f^* et g^* sont les conjuguées de Fenchel de f et g, respectivement.

Problème dual

$$\underset{u\in\mathbb{R}^{P\times L}}{\operatorname{minimize}} \left\{ \tilde{\ell}(u,\theta) := f^*(-A_\theta^* u) + g^*(u) \right\}$$

où f^* est différentiable et $g^* = \imath_{\mathcal{B}(\lambda \mathbb{1}_P)^L}$ est non différentiable

Algorithme dual forward-backward

$$\left\{\begin{array}{l} \text{for } q = 0,1,\ldots,Q-1 \\ \quad \left\lfloor\, u^{(q+1)}(\theta) = \operatorname{prox}_{\beta g^*}\left(u^{(q)}(\theta) + \beta A_\theta \nabla f^*(-A_\theta^* u^{(q)}(\theta))\right)\right. \\ w^{(Q)}(\theta) = \nabla f^*(-A_\theta u^{(Q)}(\theta)). \end{array}\right.$$

$$\boxed{\text{Cependant } \operatorname{prox}_{\beta g^*} = \operatorname{Proj}_{\mathcal{B}(\lambda)^L} \text{ est non différentiable}}$$

Algorithme dual forward-backward avec distances de Bregman

$$\left\{\begin{array}{l} \text{for } q = 0,1,\ldots,Q-1 \\ \quad \left\lfloor\, u^{(q+1)}(\theta) = \operatorname{prox}^{\Phi}_{\beta g^*}\left(\nabla\Phi(u^{(q)}(\theta)) + \beta A_\theta \nabla f^*(-A_\theta^* u^{(q)}(\theta))\right)\right. \\ w^{(Q)}(\theta) = \nabla f^*(-A_\theta u^{(Q)}(\theta)). \end{array}\right.$$

où l'opérateur proximal de Bregman avec la fonction de Legendre Φ :

$$\operatorname{prox}^{\Phi}_h(v) = \underset{u\in\mathbb{R}^{P\times L}}{\operatorname{argmin}}\, h(u) + \Phi(u) - \langle u, v\rangle$$

Problème dual

$$\underset{u\in\mathbb{R}^{P\times L}}{\text{minimize}} \left\{ \tilde{\ell}(u,\theta) := f^*(-A_\theta^* u) + g^*(u) \right\}$$

où f^* est différentiable et $g^* = \imath_{\mathcal{B}(\lambda \mathbb{1}_P)^L}$ est non différentiable

Algorithme dual forward-backward avec distances de Bregman

$$\left\{ \begin{array}{l} \text{for } q = 0,1,\ldots,Q-1 \\ \left\lfloor \; u^{(q+1)}(\theta) = \mathrm{prox}^{\Phi}_{\beta g^*}\left(\nabla\Phi(u^{(q)}(\theta)) + \beta A_\theta \nabla f^*(-A_\theta^* u^{(q)}(\theta))\right) \right. \\ w^{(Q)}(\theta) = \nabla f^*(-A_\theta u^{(Q)}(\theta)). \end{array} \right.$$

où l'opérateur proximal de Bregman avec la fonction de Legendre Φ :

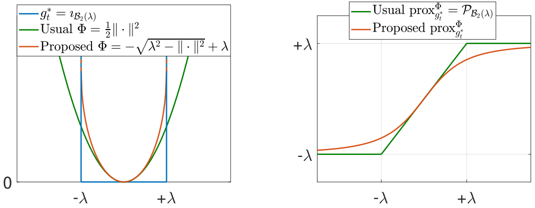
$$\mathrm{prox}_h^{\Phi}(v) = \underset{u\in\mathbb{R}^{P\times L}}{\mathrm{argmin}}\, h(u) + \Phi(u) - \langle u, v\rangle$$

Algorithme dual forward-backward avec distances de Bregman

$$\left\{\begin{array}{l} \text{for } q = 0, 1, \ldots, Q-1 \\ \left\lfloor \; u^{(q+1)}(\theta) = \mathrm{prox}^{\Phi}_{\beta g^*}\left(\nabla \Phi(u^{(q)}(\theta)) + \beta A_\theta \nabla f^*(-A^*_\theta u^{(q)}(\theta))\right) \right. \\ w^{(Q)}(\theta) = \nabla f^*(-A_\theta u^{(Q)}(\theta)). \end{array}\right.$$

où

$$\mathrm{prox}^{\Phi}_{\beta g^*}(v) = \underset{u \in \mathbb{R}^{P \times L}}{\operatorname{argmin}}\, \beta g^*(u) + \Phi(u) - \langle u, v \rangle$$



Choix de Φ pour lisser les mises à jour

Algorithme dual forward-backward avec distances de Bregman

$$\begin{cases} \text{for } q = 0, 1, \dots, Q-1 \\ \left\lfloor \; u^{(q+1)}(\theta) = \mathrm{prox}^{\Phi}_{\beta g^*}\big(\nabla\Phi(u^{(q)}(\theta)) + \beta A_\theta \nabla f^*(-A_\theta^* u^{(q)}(\theta))\big) \right. \\ w^{(Q)}(\theta) = \nabla f^*(-A_\theta u^{(Q)}(\theta)). \end{cases}$$

où

$$\mathrm{prox}^{\Phi}_{\beta g^*}(v) = \underset{u \in \mathbb{R}^{P \times L}}{\mathrm{argmin}} \sum_{l=1}^{L} \imath_{\mathcal{B}(\lambda)}(u_l) + \Phi(u) - \langle u, v \rangle$$

Algorithme dual forward-backward avec distances de Bregman

$$\left\{\begin{array}{l} \text{for } q = 0, 1, \ldots, Q-1 \\ \left\lfloor \; u^{(q+1)}(\theta) = \operatorname{prox}^{\Phi}_{\beta g^*}\left(\nabla\Phi(u^{(q)}(\theta)) + \beta A_\theta \nabla f^*(-A_\theta^* u^{(q)}(\theta))\right) \right. \\ w^{(Q)}(\theta) = \nabla f^*(-A_\theta u^{(Q)}(\theta)). \end{array}\right.$$

où

$$\operatorname{prox}^{\Phi}_{\beta g^*}(v) = \operatorname*{argmin}_{u \in \mathbb{R}^{P \times L}} \sum_{l=1}^{L} \left(\imath_{\mathcal{B}(\lambda)}(u_l) - \sqrt{\lambda^2 - \|u_l\|^2} - \langle u_l, v_l \rangle \right)$$

pour $\Phi(u) = \sum_{l=1}^{L} -\sqrt{\lambda^2 - \|u_l\|^2}$

Algorithme dual forward-backward avec distances de Bregman

$$\left\{\begin{array}{l} \text{for } q = 0, 1, \ldots, Q-1 \\ \left\lfloor\; u^{(q+1)}(\theta) = \operatorname{prox}^{\Phi}_{\beta g^*}\left(\nabla \Phi(u^{(q)}(\theta)) + \beta A_\theta \nabla f^*(-A^*_\theta u^{(q)}(\theta))\right)\right. \\ w^{(Q)}(\theta) = \nabla f^*(-A_\theta u^{(Q)}(\theta)). \end{array}\right.$$

où

$$\operatorname{prox}^{\Phi}_{\beta g^*}(v) = \left(\frac{\lambda v_l}{\sqrt{1 + \|v_l\|_2^2}}\right)_{l=1,\ldots,L}$$

for $\Phi(u) = \sum_{l=1}^{L} -\sqrt{\lambda^2 - \|u_l\|^2}$ $\quad\Rightarrow\quad$ mise à jour différentiable

[

plain,c]

Algorithmic Solution

Optimization problem

$$\underset{w\in\mathbb{R}^P}{\text{minimize}} \quad \underbrace{\frac{1}{2}\|y - Xw\|_2^2}_{f(w)\ \text{differentiable}} + \underbrace{\lambda \sum_{l=1}^{L} \|\theta_l \odot w\|_2}_{g(A_\theta w)\ \text{non differentiable}}$$

where $A_\theta : w \in \mathbb{R}^P \mapsto (\theta_1 \odot w, \ldots, \theta_L \odot w) \in \mathbb{R}^{P\times L}$

Forward-backward algorithm [Combettes and Wajs (2005)]

$$\left\{ \begin{array}{l} \text{for } i = 0, \ldots, k-1 \\ \left\lfloor \; w^{(i+1)}(\theta) = \operatorname{prox}_{\beta g \circ A_\theta}\left(w^{(i)}(\theta) - \beta \nabla f(w^{(i)}(\theta))\right) \right. \end{array} \right.$$

proximity operator (prox) ? $\longrightarrow$ *see next slide*

Optimization problem

$$\underset{w \in \mathbb{R}^P}{\text{minimize}} \quad \underbrace{\frac{1}{2}\|y - Xw\|_2^2}_{f(w) \text{ differentiable}} + \underbrace{\lambda \sum_{l=1}^{L} \|\theta_l \odot w\|_2}_{g(A_\theta w) \text{ non differentiable}}$$

where $A_\theta : w \in \mathbb{R}^P \mapsto (\theta_1 \odot w, \dots, \theta_L \odot w) \in \mathbb{R}^{P \times L}$

Forward-backward algorithm [Combettes and Wajs (2005)]

$$\begin{cases} \text{for } i = 0, \dots, k-1 \\ \quad \lfloor \; w^{(i+1)}(\theta) = \text{prox}_{\beta g \circ A_\theta}\left(w^{(i)}(\theta) - \beta \nabla f(w^{(i)}(\theta))\right) \end{cases}$$

proximity operator (prox) *?* $\longrightarrow$ *see next slide*

In the 1960s, [Moreau (1962)] proposed an extension of the notion of projection operator to any convex function h, leading to the so-called proximity operator

$$\begin{aligned}\mathrm{Proj}_{\mathcal{C}}(v) &= \underset{w\in\mathcal{C}}{\operatorname{argmin}}\ \frac{1}{2}\|w-v\|_2^2 \\ &= \underset{w\in\mathbb{R}^P}{\operatorname{argmin}}\ \imath_{\mathcal{C}}(w) + \frac{1}{2}\|w-v\|_2^2 \quad \text{where} \quad \imath_{\mathcal{C}}(w) = \begin{cases} 0 & \text{if } w\in\mathcal{C} \\ +\infty & \text{otherwise}\end{cases}\end{aligned}$$

$$\mathrm{prox}_h(v) = \underset{w\in\mathbb{R}^P}{\operatorname{argmin}}\ h(w) + \frac{1}{2}\|w-v\|_2^2$$

Optimization problem

$$\underset{w\in\mathbb{R}^P}{\text{minimize}} \; \underbrace{\frac{1}{2}\|y - Xw\|_2^2}_{f(w)\text{ differentiable}} + \underbrace{\lambda \sum_{l=1}^{L} \|\theta_l \odot w\|_2}_{g(A_\theta w)\text{ non differentiable}}$$

where $A_\theta : w \in \mathbb{R}^P \mapsto (\theta_1 \odot w, \dots, \theta_L \odot w) \in \mathbb{R}^{P\times L}$

Forward-backward algorithm [Combettes and Wajs (2005)]

$$\left\{\begin{array}{l} \text{for } i = 0, \dots, k-1 \\ \left\lfloor \; w^{(i+1)}(\theta) = \operatorname{prox}_{\beta g \circ A_\theta}\Big(w^{(i)}(\theta) - \beta \nabla f(w^{(i)}(\theta))\Big) \right. \end{array}\right.$$

generalization of projected gradient descent
projection $\longrightarrow$ *proximity operator*

$\operatorname{prox}_{\beta g\circ A_\theta}(v) = \underset{w\in\mathbb{R}^P}{\operatorname{argmin}}\; \beta g(A_\theta w) + \frac{1}{2}\|w - v\|_2^2$ ⚠ without closed form

Optimization problem

$$\underset{w\in\mathbb{R}^P}{\text{minimize}} \quad \underbrace{\frac{1}{2}\|y - Xw\|_2^2}_{f(w)\text{ differentiable}} + \underbrace{\lambda \sum_{l=1}^{L} \|\theta_l \odot w\|_2}_{g(A_\theta w)\text{ non differentiable}}$$

where $A_\theta : w \in \mathbb{R}^P \mapsto (\theta_1 \odot w, \ldots, \theta_L \odot w) \in \mathbb{R}^{P\times L}$

Forward-backward algorithm [Combettes and Wajs (2005)]

$$\left\{ \begin{array}{l} \text{for } i = 0, \ldots, k-1 \\ \quad \left\lfloor \; w^{(i+1)}(\theta) = \mathrm{prox}_{\beta g \circ A_\theta}\left(w^{(i)}(\theta) - \beta \nabla f(w^{(i)}(\theta))\right) \right. \end{array} \right.$$

generalization of projected gradient descent
projection $\longrightarrow$ proximity operator

$$\mathrm{prox}_{\beta g\circ A_\theta}(v) = \underset{w\in\mathbb{R}^P}{\mathrm{argmin}}\; \beta g(A_\theta w) + \tfrac{1}{2}\|w - v\|_2^2$$ ⚠ without closed form

The ideas of duality and transforms are ubiquitous in mathematics

- Harmonics analysis $\rightarrow$ Fourier transform
- Convex analysis $\rightarrow$ Fenchel conjugate : $h^*(x) = \sup_w \langle w, x \rangle - h(w)$

[Rockafellar (1970)]

Example : $h\colon w \mapsto \|w\|_2 \implies h^*: x \mapsto \imath_{\mathcal{B}(1)}(x) = \begin{cases} 0 & \text{if } \|x\|_2 \leq 1 \\ +\infty & \text{otherwise} \end{cases}$

The ideas of duality and transforms are ubiquitous in mathematics

- Harmonics analysis $\rightarrow$ Fourier transform
- Convex analysis $\rightarrow$ Fenchel conjugate : $h^*(x) = \sup_w \langle w, x\rangle - h(w)$

[Rockafellar (1970)]

Example : $h\colon w \mapsto \lambda\|w\|_2 \quad \Rightarrow \quad h^*: x \mapsto \imath_{\mathcal{B}(\lambda)}(x) = \begin{cases} 0 & \text{if } \|x\|_2 \leq \lambda \\ +\infty & \text{otherwise} \end{cases}$

Duality in convex optimization

The ideas of duality and transforms are ubiquitous in mathematics

- Harmonics analysis $\rightarrow$ Fourier transform
- Convex analysis $\rightarrow$ Fenchel conjugate : $h^*(x) = \sup_w \langle w, x \rangle - h(w)$

[Rockafellar (1970)]

Primal problem $\longleftrightarrow$ **Dual problem**

$$\underset{w \in \mathbb{R}^P}{\text{minimize}}\ f(w) + g(A_\theta w)$$

$$A_\theta : \mathbb{R}^P \rightarrow \mathbb{R}^{P \times L}$$

$$g(v_1 \ldots v_L) = \sum_{l=1}^{L} \underbrace{\lambda \|v_l\|_2}_{\text{norm}}$$

$$\underset{u \in \mathbb{R}^{P \times L}}{\text{minimize}}\ f^*(-A_\theta^\top u) + g^*(u)$$

$$A_\theta^\top : \mathbb{R}^{P \times L} \rightarrow \mathbb{R}^P$$

$$g^*(u_1 \ldots u_L) = \sum_{l=1}^{L} \underbrace{\imath_{\mathcal{B}(\lambda)}(u_l)}_{\substack{\text{indicator}\\ \text{dual norm ball}}}$$

Link

$$w = \nabla f^*(-A_\theta^\top u)$$

$\boxed{\mathrm{prox}_{g \circ A_\theta} \text{ without closed form} \Rightarrow \text{solve dual problem to move } A_\theta \text{ in smooth part}}$

Dual problem

$$\underset{u \in \mathbb{R}^{P \times L}}{\text{minimize}} \; \underbrace{f^*(-A_\theta^\top u)}_{\text{differentiable}} + \underbrace{g^*(u)}_{\text{non differentiable}}$$

Dual forward-backward algorithm

$$\begin{cases} \text{for } i = 0, \ldots, k-1 \\ \left\lfloor \; u^{(i+1)}(\theta) = \mathrm{prox}_{\beta g^*}\left(u^{(i)}(\theta) + \beta A_\theta \nabla f^*(-A_\theta^\top u^{(i)}(\theta))\right) \right. \\ w^{(k)}(\theta) = \nabla f^*(-A_\theta^* u^{(k)}(\theta)) \qquad \textbf{(link)} \end{cases}$$

Dual problem

$$\underset{u\in\mathbb{R}^{P\times L}}{\text{minimize}} \; \underbrace{f^*(-A_\theta^\top u)}_{\text{differentiable}} + \underbrace{g^*(u)}_{\text{non differentiable}}$$

Dual forward-backward algorithm

$$\begin{cases} \text{for } i = 0, \ldots, k-1 \\ \quad \lfloor \; u^{(i+1)}(\theta) = \operatorname{prox}_{\beta g^*}\left(u^{(i)}(\theta) + \beta A_\theta \nabla f^*(-A_\theta^\top u^{(i)}(\theta))\right) \\ w^{(k)}(\theta) = \nabla f^*(-A_\theta^* u^{(k)}(\theta)) \qquad \textbf{(link)} \end{cases}$$

where the proximal operator reads :

$$\begin{aligned} \operatorname{prox}_{\beta g^*}(v) &= \underset{u\in\mathbb{R}^{P\times L}}{\operatorname{argmin}} \, \beta g^*(u) + \frac{1}{2}\|u - v\|^2 \\ &= \underset{u\in\mathbb{R}^{P\times L}}{\operatorname{argmin}} \, \beta \sum_{l=1}^{L} \imath_{\mathcal{B}(\lambda)}(u_l) + \frac{1}{2}\|u - v\|^2 \\ &= \operatorname{Proj}_{\mathcal{B}(\lambda)^L}(v) \quad ⚠ \; \text{not differentiable} \end{aligned}$$

Reminder : why differentiability is important

$\theta^{(0)}$ chosen arbitrarily
for $n = 0, 1, \dots$

$$
\left\lfloor
\begin{array}{l}
w^{(0)}(\theta^{(n)}) \text{ chosen arbitrarily} \\
\text{for } i = 0, \dots, k-1 \qquad\qquad\qquad\qquad \text{(inner algorithm = dual forward-backward)} \\
\left\lfloor \; w^{(i+1)}(\theta^{(n)}) = \mathcal{A}(w^{(i)}(\theta^{(n)})) \right. \\
\theta^{(n+1)} = \mathrm{Proj}_\Theta(\theta^{(n)} - \gamma \nabla \mathcal{U}^{(k)}(\theta^{(n)})) \quad \text{where} \quad \mathcal{U}^{(k)}(\theta^{(n)}) = \mathcal{E}(w^{(k)}(\theta^{(n)}))
\end{array}
\right.
$$

We want a differentiable dual forward-backward algorithm
because it inside a bilevel algorithm !

Dual forward-backward algorithm

$$\begin{cases} \text{for } i = 0, \dots, k-1 \\ \left\lfloor \; u^{(i+1)}(\theta) = \operatorname{prox}_{\beta g^*}\Big(\quad u^{(i)}(\theta) \;\; + \beta A_\theta \nabla f^*(-A_\theta^\top u^{(i)}(\theta))\Big) \right. \\ w^{(k)}(\theta) = \nabla f^*(-A_\theta u^{(k)}(\theta)). \end{cases}$$

where

$$\begin{aligned} \operatorname{prox}_{\beta g^*}(v) &= \underset{u \in \mathbb{R}^{P \times L}}{\operatorname{argmin}} \, \beta g^*(u) + \frac{1}{2}\|u - v\|^2 \\ &= \underset{u \in \mathbb{R}^{P \times L}}{\operatorname{argmin}} \, \beta g^*(u) + \frac{1}{2}\|u\|^2 - \langle u, v \rangle + \mathrm{cst} \end{aligned}$$

Dual forward-backward algorithm

$$\begin{cases} \text{for } i = 0, \dots, k-1 \\ \left\lfloor \; u^{(i+1)}(\theta) = \mathrm{prox}_{\beta g^*}\big(\quad u^{(i)}(\theta) \; + \beta A_\theta \nabla f^*(-A_\theta^\top u^{(i)}(\theta))\big) \right. \\ w^{(k)}(\theta) = \nabla f^*(-A_\theta u^{(k)}(\theta)). \end{cases}$$

where

$$\mathrm{prox}_{\beta g^*}(v) = \underset{u \in \mathbb{R}^{P \times L}}{\mathrm{argmin}}\, \beta g^*(u) + \frac{1}{2}\|u\|^2 - \langle u, v \rangle + \mathrm{cst}$$

Group Lasso solver $\mathcal{A}$: dual approach

Dual forward-backward algorithm with Bregman distances [Bauschke et al. (2016)]

$$\begin{cases} \text{for } i = 0, \ldots, k-1 \\ \quad \lfloor \; u^{(i+1)}(\theta) = \mathrm{prox}^{\Phi}_{\beta g^*}\Big(\nabla\Phi(u^{(i)}(\theta)) + \beta A_\theta \nabla f^*(-A_\theta^\top u^{(i)}(\theta))\Big) \\ w^{(k)}(\theta) = \nabla f^*(-A_\theta u^{(k)}(\theta)). \end{cases}$$

where the Bregman proximal operator associated to Φ :

$$\mathrm{prox}^{\Phi}_{\beta g^*}(v) = \underset{u \in \mathbb{R}^{P \times L}}{\mathrm{argmin}}\, \beta g^*(u) + \Phi(u) - \langle u, v \rangle$$

Dual forward-backward algorithm with Bregman distances

$$\begin{cases} \text{for } i = 0, \ldots, k-1 \\ \left\lfloor \; u^{(i+1)}(\theta) = \mathrm{prox}^{\Phi}_{\beta g^*}\big(\nabla\Phi(u^{(i)}(\theta)) + \beta A_\theta \nabla f^*(-A_\theta^\top u^{(i)}(\theta))\big) \right. \\ w^{(k)}(\theta) = \nabla f^*(-A_\theta u^{(k)}(\theta)). \end{cases}$$

where

$$\mathrm{prox}^{\Phi}_{\beta g^*}(v) = \underset{u \in \mathbb{R}^{P \times L}}{\mathrm{argmin}}\, \beta g^*(u) + \Phi(u) - \langle u, v \rangle$$

Dual forward-backward algorithm with Bregman distances

$$\begin{cases} \text{for } i = 0, \ldots, k-1 \\ \left\lfloor \; u^{(i+1)}(\theta) = \mathrm{prox}^{\Phi}_{\beta g^*}\big(\nabla\Phi(u^{(i)}(\theta)) + \beta A_\theta \nabla f^*(-A_\theta^\top u^{(i)}(\theta))\big) \right. \\ w^{(k)}(\theta) = \nabla f^*(-A_\theta u^{(k)}(\theta)). \end{cases}$$

where

$$\mathrm{prox}^{\Phi}_{\beta g^*}(v) = \underset{u \in \mathbb{R}^{P \times L}}{\mathrm{argmin}} \sum_{l=1}^{L} \imath_{\mathcal{B}(\lambda)}(u_l) + \Phi(u) - \langle u, v \rangle$$

Dual forward-backward algorithm with Bregman distances

$$\begin{cases} \text{for } i = 0, \ldots, k-1 \\ \quad \left\lfloor \; u^{(i+1)}(\theta) = \operatorname{prox}^{\Phi}_{\beta g^*}\big(\nabla\Phi(u^{(i)}(\theta)) + \beta A_\theta \nabla f^*(-A_\theta^\top u^{(i)}(\theta))\big) \right. \\ w^{(k)}(\theta) = \nabla f^*(-A_\theta u^{(k)}(\theta)). \end{cases}$$

where

$$\operatorname{prox}^{\Phi}_{\beta g^*}(v) = \operatorname*{argmin}_{u \in \mathbb{R}^{P \times L}} \sum_{l=1}^{L} \big(\imath_{\mathcal{B}(\lambda)}(u_l) + \phi(u_l) - \langle u_l, v_l \rangle\big)$$

for $\Phi(u) = \sum_{l=1}^{L} \phi(u_l)$

Dual forward-backward algorithm with Bregman distances

$$\begin{cases} \text{for } i = 0, \dots, k-1 \\ \quad \big\lfloor \; u^{(i+1)}(\theta) = \mathrm{prox}^{\Phi}_{\beta g^*}\big(\nabla\Phi(u^{(i)}(\theta)) + \beta A_\theta \nabla f^*(-A_\theta^\top u^{(i)}(\theta))\big) \\ w^{(k)}(\theta) = \nabla f^*(-A_\theta u^{(k)}(\theta)). \end{cases}$$

where

$$\mathrm{prox}^{\Phi}_{\beta g^*}(v) = \underset{u \in \mathbb{R}^{P\times L}}{\mathrm{argmin}} \sum_{l=1}^{L} \Big(\imath_{\mathcal{B}(\lambda)}(u_l) - \sqrt{\lambda^2 - \|u_l\|^2} - \langle u_l, v_l \rangle \Big)$$

for $\phi(u_l) = -\sqrt{\lambda^2 - \|u_l\|^2} \Rightarrow \mathrm{dom}\,\phi = \mathcal{B}(\lambda)$

$\Rightarrow \imath_{\mathcal{B}(\lambda)}(u_l)$ always equal to 0 !

⚠ **trick for a differentiable algorithm**

Dual forward-backward algorithm with Bregman distances

$$\begin{cases} \text{for } i = 0, \ldots, k-1 \\ \left\lfloor \; u^{(i+1)}(\theta) = \mathrm{prox}^{\Phi}_{\beta g^*}\left(\nabla\Phi(u^{(i)}(\theta)) + \beta A_\theta \nabla f^*(-A_\theta^\top u^{(i)}(\theta))\right) \right. \\ w^{(k)}(\theta) = \nabla f^*(-A_\theta u^{(k)}(\theta)). \end{cases}$$

where

$$\mathrm{prox}^{\Phi}_{\beta g^*}(v) = \left(\frac{\lambda v_l}{\sqrt{1 + \|v_l\|_2^2}}\right)_{l=1,\ldots,L}$$