UNLocBoX : Une boîte à outils MATLAB pour l'optimisation convexe

Mise en œuvre efficace des méthodes d'éclatement proximal

KUITCHE AROLLE NACHARD 22T2931
KONZOU SODEA ALAN PEREC 22T2957
KENFACK LEKANE FRANK 22T2847
VOUKENG DJIOKENG CHRISTIAN ROUSSEL 22U2053

Département d'Informatique

Faculté des Sciences

Sommaire

Le problème cible

Résoudre des problèmes de minimisation de la forme :

$$\min_{x \in \mathbb{R}^N} \sum_{n=1}^{K} f_n(x)$$

- De nombreux problèmes en Machine Learning, traitement du signal et imagerie peuvent être formulés ainsi.
- **Idée clé :** La fonction objectif est une somme de termes "simples" f_n.

Qu'est-ce que UNLocBoX ?

- Une bibliothèque MATLAB open-source pour l'optimisation convexe.
- **Focus :** Les méthodes de "proximal splitting" (éclatement proximal).
- **Objectifs de conception :**
  - **Simple** pour les novices (choix automatique du solveur).
  - **Flexible** pour les experts (contrôle total des paramètres).
  - **Efficace et scalable** (adaptée aux "Big Data").
- **Positionnement (vs CVX) :**
  - CVX est une "boîte noire" plus simple.
  - UNLocBoX est beaucoup plus efficace et scalable.

Le concept : "Proximal Splitting"

- **Principe :** Résoudre des sous-problèmes plus simples basés sur chaque f_n séparément, de manière itérative.
- **Exemple (LASSO) :**

$$\min_x \underbrace{||Ax - y||_2^2}_{f_1 \text{ (lisse)}} + \underbrace{\gamma||x||_1}_{f_2 \text{ (non-lisse)}}$$

- **Algorithme (Forward-Backward) :**
  1. Faire un pas de gradient sur f_1 (partie lisse).
  2. Faire un pas "proximal" sur f_2 (partie non-lisse).

L'Opérateur Proximal : La brique de base

Définition

$$\text{prox}_f(x) := \arg\min_y \left\{ \frac{1}{2}||x - y||_2^2 + f(y) \right\}$$

- **Intuition :** Le proximal de f trouve un point y proche de x qui minimise f.
- **Rôle :** Il généralise l'étape de gradient pour les fonctions non différentiables.
- **Exemple (Norme l_1) :** Le proximal de $\gamma||x||_1$ est l'opérateur de seuillage doux (soft-thresholding).

Installation et Démarrage

Téléchargement

- Site web : `https://lts2.epfl.ch/unlocbox/`
- Via Git (version de développement) :

```
git clone https://github.com/nperraud/unlocbox.git
```

Initialisation

Lancer `init_unlocbox.m` dans MATLAB pour ajouter les chemins.

- **Dépendances (Optionnelles) :** LTFAT, GSPBox.

Structure de la Boîte à Outils

UNLocBoX est composée de 4 parties principales :

1. **Solvers :** Le cœur. Les algorithmes (ex : `forward_backward`).
2. **Opérateurs Proximaux :** Implémentation des proximaux (ex : `prox_l1`, `prox_tv`).
3. **Fichiers de Démonstration :** Exemples d'utilisation.
4. **Fonctions Utilitaires :** Fonctions diverses.

Comment l'utiliser : 2 Étapes

1. **Définir le problème :**
   Modéliser chaque fonction f_n comme une structure MATLAB.
2. **Choisir un solveur :**
   Lancer un solveur (ou laisser `solvep` choisir) pour minimiser la somme des fonctions.

Étape 1 : Définir les Fonctions (Structure `f`)

- Chaque fonction f_n est une 'struct'.
- Elle doit contenir certains champs selon sa nature.
- **Champ commun (Optionnel) :**
  - `f.eval = @(x) ...` : Une fonction qui retourne la valeur $f(x)$.

Cas 1 : Fonctions Différentiables (Lisses)

Pour les fonctions lisses (ex : $||Ax - y||_2^2$).

Champs requis

- `f.grad = @(x) ...` : Handle vers la fonction calculant le gradient $\nabla f(x)$.
- `f.beta = ...` : Une borne supérieure sur la constante de Lipschitz du gradient.

Exemple : $f(x) = 5||Ax - y||_2^2$

```
f.eval = @(x) 5 * norm(A*x - y)^2;
f.grad = @(x) 2 * 5 * A' * (A*x - y);
f.beta = 2 * 5 * norm(A)^2;
```

Cas 2 : Fonctions Non Différentiables (Non-lisses)

Pour les fonctions non-lisses (ex : $||x||_1$, $||x||_{TV}$).

Champ requis

- `f.prox = @(x, T) ...` : Handle vers la fonction calculant l'opérateur proximal $\text{prox}_{Tf}(x)$.

Exemple : $f(x) = 7||x||_1$

On utilise le proximal `prox_l1` fourni par la toolbox.

```
f.eval = @(x) 7 * norm(x, 1);
% On passe le poids (7) au proximal
f.prox = @(x, T) prox_l1(x, 7 * T);
```

Cas 3 : Les Contraintes (Projections)

- Une contrainte $x \in \mathcal{C}$ est gérée via sa **fonction indicatrice** $i_{\mathcal{C}}(x)$.

$$i_{\mathcal{C}}(x) = \begin{cases} 0 & \text{si } x \in \mathcal{C} \\ +\infty & \text{sinon} \end{cases}$$

- **Le proximal d'une fonction indicatrice est une projection !**

Exemple : Contrainte $||x||_2 \leq 2$

```
% On utilise prox_b2 (projection sur la boule L2)
param_b2.epsilon = 2;
f.prox = @(x, T) prox_b2(x, T, param_b2);
f.eval = @(x) 0; % La valeur n'importe pas
```

Étape 2 : Lancer le Solveur

La fonction principale : `solvep`

- `solvep` analyse les champs (`grad`, `prox`) des fonctions fournies et **choisit automatiquement le meilleur solveur**.

Syntaxe

```
sol = solvep(x_0, {f1, f2, f3}, param);
```

- x_0 : Point d'initialisation.
- $\{f_1, f_2, f_3\}$: Cell array contenant les structures de vos fonctions.
- `param` : Structure optionnelle pour les paramètres.

Les Solvers Disponibles

UNLocBoX implémente les algorithmes "state-of-the-art".

Solvers spécifiques (2 fonctions)

- `forward_backward` (FISTA)
  (1 lisse + 1 non-lisse)
- `douglas_rachford`
  (2 non-lisses)
- Primal-Dual (ADMM, ...)

Solvers généraux ($>$ 2 fonctions)

- `generalized_forward_backward`
- `sdmm`

Paramètres Optionnels (param)

Permet de contrôler finement l'exécution du solveur.

Paramètres communs

```
param.maxit = 100;      % Nombre max d'itérations
param.tol = 1e-5;       % Tolérance d'arrêt
param.verbose = 1;      % Niveau de verbosité: 0, 1, 2
param.gamma = ...;      % Taille de pas (si non auto)
```

Plug-ins et Sorties

Plug-ins : `param.do_sol`

Permet d'exécuter une fonction à chaque itération.

```
% Affiche l'image 'x' à chaque 10 itérations
fig = figure(100);
param.do_sol = @(x) plot_image(x, fig);
param.period = 10;
```

Arguments de sortie

```
[sol, info] = solvep(...)
```

- `sol` : La solution finale (le minimiseur).
- `info` : Structure avec infos de convergence (itérations, erreur, ...).

Exemple Complet : LASSO

Problème :

$$\min_x ||Ax - y||_2^2 + \lambda ||x||_1$$

Code UNLocBoX ($\lambda = 0.1$)

```
% f1: Terme de fidélité (lisse)
f1.eval = @(x) norm(A*x - y)^2;
f1.grad = @(x) 2 * A' * (A*x - y);
f1.beta = 2 * norm(A)^2;

% f2: Régularisation L1 (non-lisse)
f2.eval = @(x) lambda * norm(x, 1);
f2.prox = @(x, T) prox_l1(x, lambda * T);

% Initialisation et solveur
x0 = zeros(N, 1);
```

Conclusion

- UNLocBoX est une boîte à outils puissante et flexible pour l'optimisation convexe.
- **Philosophie :** Décomposer un problème complexe en une somme de fonctions simples.
- **Mise en œuvre :** Définir chaque fonction f_n via ses briques d'analyse convexe (gradient ou proximal).
- **Exécution :** `solvep` s'occupe de sélectionner l'algorithme de "proximal splitting" approprié.

Questions ?

Ressources

- Documentation officielle :
  `https://lts2.epfl.ch/unlocbox/doc`
- Dépôt GitHub :
  `https://github.com/epfl-lts2/unlocbox/releases/tag/1.7.5`

Merci de votre attention !