UNLocBoX : Une Boîte à Outils MATLAB pour l'Optimisation Convexe

MEFFO Lea 22U2194
KEMBOU Richel 22U2118
KELODJOU Ivana 22T2894
TSEMEGNE Martin 22U2080

Université de Yaoundé I

(jecy.meffo, richel.kembou, ivana.kelodjou, yvan.tsemegne)@facsciences-uy1.cm

October 21, 2025

SOMMAIRE

Contexte et Présentation de UNLocBoX

Le Problème d'Optimisation et le Défi

De nombreux problèmes en Machine Learning se ramènent à minimiser une fonction objectif, souvent une somme de termes simples:

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

Face au Big Data, les méthodes doivent être **efficaces** et **scalables**.

La Solution : UNLocBoX

UNLocBoX est une boîte à outils **MATLAB** dédiée à l'optimisation convexe.

Présentation de UNLocBoX

Qu'est-ce que UNLocBoX ?

Une boîte à outils (*toolbox*) **MATLAB** dédiée à l'optimisation convexe.

- **Objectif Principal :** Résoudre des problèmes d'optimisation en utilisant des méthodes de **Proximal Splitting** (décomposition proximale).
- **Philosophie :** Rester proche de la formulation mathématique du problème.
- **Public Cible :**
  - **Novice :** Définir les fonctions, un solveur général choisit la meilleure méthode.
  - **Avancé :** Contrôle total sur le choix du solveur et des paramètres.

Installation et Paramètres

Installation et Démarrage

1. Télécharger Unlocbox depuis github via le lien https://github.com/epfl-lts2/unlocbox
2. Extraire l'archive.
3. Lancer la fonction `init_unlocbox.m` au début de votre script.

Optionnel : L'utilisation du GPU est possible pour accélérer certaines opérations (ex: TV) en modifiant une variable globale.

Paramètres Optionnels Clés

La structure `param` permet de contrôler le solveur :

- `param.maxit` : Nombre maximum d'itérations.
- `param.tol` : Tolérance pour le critère d'arrêt.
- `param.gamma` : Taille du pas (step-size), si non calculée automatiquement.

Principe du Proximal Splitting

Pourquoi "Splitting" (Décomposer) ?

L'idée est de décomposer une fonction objectif complexe $f(x)$ en une somme de termes $f_n(x)$ plus simples, que l'on peut traiter séparément.

$$f(x) = f_1(x) + f_2(x) + \cdots + f_K(x)$$

Avantage Majeur

Cette décomposition permet d'utiliser des algorithmes avec une faible complexité par itération (souvent en $\mathcal{O}(N)$), ce qui est crucial pour les grands problèmes.

Le Cas des Fonctions Non-Différentiables

Fonctions Lisses

On utilise la **Descente de Gradient**.

$$x_{k+1} = x_k - \gamma \nabla f(x_k)$$

Fonctions Non-Lisses

Le gradient n'est pas défini partout.
On le remplace par l'opérateur **proximal**.

$$x_{k+1} = \text{prox}_{\gamma f}(x_k)$$

Un exemple typique de fonction non-lisse est la norme ℓ_1 ($|x|$), utilisée pour promouvoir la sparsité.

L'Opérateur Proximal : Définition

Définition Mathématique

L'opérateur proximal d'une fonction f est la solution d'un problème de minimisation :

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

Intuitivement, il trouve un point y proche de x qui minimise aussi la fonction f.

L'Opérateur Proximal : Définition

Définition Mathématique

L'opérateur proximal d'une fonction f est la solution d'un problème de minimisation :

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

Intuitivement, il trouve un point y proche de x qui minimise aussi la fonction f.

- **Propriétés Clés :**
  - Existence et unicité de la solution pour f convexe.
  - L'itération $x_{k+1} = \text{prox}_f(x_k)$ fait converger vers le minimum de f.

Exemple : Le Soft-Thresholding

Opérateur Proximal de la Norme ℓ_1

Pour la fonction $f(x) = \lambda \|x\|_1$, son opérateur proximal est une opération très connue et efficace appelée le **soft-thresholding** (seuillage doux).

Pour chaque composante α du vecteur x, l'opération est :

$$\text{soft}_\lambda(\alpha) = \begin{cases} \alpha - \lambda & \text{si } \alpha > \lambda \\ \alpha + \lambda & \text{si } \alpha < -\lambda \\ 0 & \text{sinon} \end{cases}$$

Efficacité

Cette opération est non-itérative et se calcule en $\mathcal{O}(N)$, ce qui la rend très rapide.

Architecture Globale

Les 4 Groupes de Fonctions

UNLocBoX est organisé autour de quatre composants principaux :

1. **Solvers (Solveurs) :** Le cœur de la toolbox. Contient les algorithmes d'optimisation (Forward-Backward, Douglas-Rachford, etc.).
2. **Proximal Operators (Opérateurs Proximaux) :** Une large collection de prox pré-implémentés pour les fonctions communes (normes, contraintes...).
3. **Fichiers de Démonstration :** Exemples concrets pour démarrer rapidement.
4. **Fonctions Utilitaires :** Fonctions d'aide diverses.

Étape 1 : Modélisation des Fonctions

Le concept de structure MATLAB

Dans UNLocBoX, chaque terme $f_k(x)$ de la somme est représenté par une **structure** MATLAB qui peut contenir jusqu'à quatre champs principaux.

- `f.eval` : Une fonction pour évaluer $f_k(x)$.
- `f.grad` : Une fonction pour calculer le gradient $\nabla f_k(x)$ (si différentiable).
- `f.prox` : Une fonction pour appliquer l'opérateur proximal (si non-différentiable).
- `f.beta` : La constante de Lipschitz du gradient (si différentiable).

Cas 1 : Fonctions Différentiables

Exemple : $f(x) = \frac{1}{2}\|Ax - y\|_2^2$

Cette fonction est lisse (différentiable). On doit donc définir son évaluation et son gradient.

- **Champ `f.eval` :**
  - Contient le handle de la fonction : `@(x) 0.5 * norm(A*x - y)^2`
- **Champ `f.grad` :**
  - Contient le handle du gradient : `@(x) A'*(A*x - y)`
- **Champ `f.beta` :**
  - Contient la constante de Lipschitz : `norm(A)^2`

Cas 2 : Fonctions Proximales (Non-Lisses)

Exemple : $f(x) = \lambda \|x\|_1$

Cette fonction est non-lisse. On doit définir son évaluation et son opérateur proximal.

- **Champ `f.eval` :**
  - Contient le handle de la fonction : `@(x) lambda * norm(x, 1)`
- **Champ `f.prox` :**
  - Fait appel à un opérateur pré-implémenté : `@(x, T) prox_l1(x, lambda*T)`

UNLocBoX fournit des opérateurs pour de nombreuses normes : ℓ_1, variation totale (TV), norme nucléaire, etc..

Gestion des Contraintes

Comment ajouter une contrainte $x \in \mathcal{C}$?

Une contrainte sur un ensemble convexe $\mathcal{C}$ est modélisée via la **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 de l'Indicatrice est une Projection

L'opérateur proximal de $i_{\mathcal{C}}(x)$ est simplement la **projection** sur l'ensemble $\mathcal{C}$. UNLocBoX fournit de nombreux projecteurs (boule ℓ_2, contraintes de positivité, etc.).

La Fonction Centrale : `solvep`

Le Solveur Automatique

`solvep` est la fonction principale qui simplifie l'utilisation de la toolbox. Elle analyse les fonctions fournies et sélectionne automatiquement le solveur le plus adapté.

Syntaxe

L'appel est très simple : `sol = solvep(x0, {f1, f2, f3}, param);`

- `x0` : point d'initialisation.
- `{f1, f2, ...}` : ensemble des fonctions (structures) à minimiser.
- `param` : structure optionnelle pour les paramètres.

Vue d'ensemble des Solvers

Les algorithmes de la toolbox se classent en deux catégories :

Solvers Spécifiques

Optimisés pour des problèmes avec peu de fonctions (souvent 2 ou 3).

- Forward-Backward (FISTA)
- Douglas-Rachford
- ADMM
- Chambolle-Pock

Solvers Généraux

Plus flexibles, pour des sommes avec un grand nombre de fonctions.

- PPXA
- SDMM

Cas d'Étude : Reconstruction d'Image

Problème

Reconstruire une image de haute qualité à partir d'une version bruitée et avec des pixels manquants (inpainting).

$$\textit{[Image originale]} \rightarrow \textit{[Image dégradée]} \rightarrow \textit{[Image reconstruite]}$$

Formulation Mathématique

On cherche une image x qui minimise sa Variation Totale (pour préserver les contours) tout en restant fidèle aux mesures y :

$$\min_x \|x\|_{TV} \quad \text{sujet à} \quad \|Ax - y\|_2 \leq \epsilon$$

Où A est un opérateur de masquage.

Implémentation Simplifiée

Le problème précédent se traduit en deux fonctions pour UNLocBoX :

Fonction 1 : La régularisation

$f_1(x) = \|x\|_{TV}$

- On utilise l'opérateur proximal pré-implémenté `prox_tv`.

Fonction 2 : La contrainte

$f_2(x) = i_{\mathcal{C}}(x)$ avec $\mathcal{C} = \{x \mid \|Ax - y\|_2 \leq \epsilon\}$

- On utilise le projecteur sur la boule ℓ_2, `proj_b2`.

Appel au Solveur

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

Conclusion

Pourquoi utiliser UNLocBoX ?

- **Rapidité :** Accès à des algorithmes d'optimisation de pointe, rapides et scalables.
- **Modularité :** La décomposition du problème en fonctions simples rend le code facile à lire, à maintenir et à adapter.
- **Flexibilité :** Grand choix de solveurs et d'opérateurs pour une large gamme de problèmes d'optimisation convexe.

Un outil puissant pour passer rapidement de la théorie mathématique à l'implémentation pratique.