Compressed Sensing

Relaxation Biconvexe pour de la Programmation Semidefinie en Vision par Ordinateur

---

AXEL MARCHAND, RODRIGUE RILLARDON

1 Introduction

De nombreux problèmes de vision par ordinateur peuvent être modélisés comme des problèmes binaires quadratiques.
La programmation semi-définie positive (SDP) est particulièrement utile pour approcher la solution de ce genre de problèmes, mais en général, les solutions pour les problème SDP en grande dimension sont très lentes et coûteuses en mémoire. Certains problèmes de vision par ordinateur font appel à des matrices de petit rang, mais cette contrainte de rang est une contrainte non convexe, ce qui rend la recherche de solution optimale plus compliquée. L'avantage des méthodes SDP[4], par rapport aux méthodes spectrales, est qu'elles permettent d'obtenir des approximation plus fines de la solution optimale. En revanche, elles sont peu scalables et adaptables aux problèmes en grandes dimensions.
Le problème d'optimisation non convexe peuvent être résolus grâce à la relaxation semi-definie (SDR). Utiliser la SDR augmente de manière significative la dimension du problème initial, notamment en transformant les vecteurs en matrices, ce qui augmente de manière conséquente les coûts de calcul et la mémoire nécessaire. Le but de l'article étudié est de présenter une méthode de résolution des problèmes de programmation SDP, sous contrainte d'utilisation de matrices semi-définies positives, de manière rapide et peu coûteuse en mémoire. Nous présenterons ainsi, la relaxation bi-convexe (BCR)[1], qui transforme un problème SDP en un problème d'optimisation bi-convexe. Ce problème d'optimisation bi-convexe peut être résolu en utilisant une procédure AM (Alternating Minimization Algorithm), qui est peu coûteuse en mémoire et en temps de calcul (que l'on présentera dans la suite).

2 Rappel sur la programmation SDP

Les problèmes de programmation SDP [4] s'écrivent sous la forme suivante :

$$\begin{aligned} & min_{Y \in S^{+}_{NXN}} < C, Y > \\ s.c & < A_i, Y >= b_i, \forall i \in \mathcal{E} \\ & < A_j, Y > \leqslant b_j, \forall j \in \mathcal{B} \end{aligned} \tag{1}$$

Avec S^{+}_{NXN} l'ensemble des matrices semi-définies positives de taille NxN et, $< C, Y >= tr(C^T Y)$. On notera que les ensembles $\mathcal{E}$ et $\mathcal{B}$ contiennent respectivement les indices associés aux contraintes d'égalité et d'inégalité, et que les matrices A_i et A_j sont des matrices symétriques.
Il y a deux avantages principaux à la programmation SDP :

— elle permet de transformer certaines contraintes non-convexe en des contraintes convexes grâce à la relaxation semi-définie

— elle garantit des résultats théoriques importants

Ainsi, dans le domaine de la vision par ordinateur, de nombreux problèmes peuvent être modélisés sous la forme (1), par exemple dans le couplage pour la théorie des graphes ou en reconnaissance d'images.

La relaxation semi-définie est assez utile pour résoudre des problèmes de labellisation binaire. Dans ce genre de problème, on cherche à minimiser une fonction de perte quadratique. Les problèmes binaires quadratiques (BQP) sont par exemple très utilisés pour la segmentation d'images, le couplage ou la restauration d'images. Les problèmes BQP peuvent être résolus en transformant un vecteur $b \in \{\pm 1\}^N$ en une matrice semi-définie positive $B = b^T b$, et dont la contrainte de rang est transformée en une contrainte de semi-définie positivité. Ainsi par exemple, le problème BQP suivant :

$$\begin{aligned} & min_{\mathrm{x}} \mathrm{x}^T A \mathrm{x} \\ & s.c\ \mathrm{x} \in \{\pm 1\}^N \end{aligned} \tag{2}$$

peut être résolu grâce au problème de relaxation semi-définie (SDR) suivant :

$$\begin{aligned} & min_{X \in S^+_{NXN}} < X, A > \\ & s.c\ \mathrm{diag}(X) = e, rank(X) = 1 \end{aligned} \tag{3}$$

où $X = \mathrm{x}^T \mathrm{x}$. Un autre avantage de la programmation SDP est de pouvoir être adaptable à des problèmes de différentes formes.

3 La relaxation bi-convexe (BCR)

3.1 Présentation de la relaxation bi-convexe

Plutôt que des résoudre (1), on peut utiliser le fait que toute matrice symétrique semi-définie positive peut s'écrire comme $Y = X^T X$, et donc en remplaçant dans 1, on obtient :

$$\begin{aligned} & min_{X \in \mathbb{R}^{NXr}}\ \mathrm{tr}(X^T C X) \\ & s.c\ \mathrm{tr}(X^T A_i X) = b_i, \forall i \in \mathcal{E} \\ & tr(X^T A_i X) \leqslant b_j, \forall j \in \mathcal{B} \end{aligned} \tag{4}$$

où r = rang(Y). De plus, comme pour Y, on peut réécrire A sous la forme $A = L^T L$ et utiliser $tr(X^T AX) = tr(X^T L^T LX) = ||LX||_F^2$ (où $||.||_F^2$ est la norme Frobenius). On peut alors réécrire le problème sous une autre forme :

$$\begin{aligned} & min_{X \in \mathbb{R}^{NXr}} \text{ tr}(X^T CX) \\ \text{s.c } & Q_i = L_i X, ||Q_i||_F^2 = b_i, \forall i \in \mathcal{E} \\ & Q_j = L_j X, ||Q_j||_F^2 \leqslant b_j, \forall j \in \mathcal{B} \end{aligned} \tag{5}$$

Si les matrices A_i, A_j et C sont elles-mêmes semi-définies positives, alors la fonction objectif dans (5) est convexe et quadratique et les contraintes d'inégalité sont également convexes. La non-convexité du problème vient simplement des contraintes d'égalité. On va alors chercher à transformer ces contraintes d'égalité.
Dans la formulation (5) du problème, on n'a plus de convexité. Toutefois, si r < N, alors la dimension du problème est grandement réduite par rapport à (1). On va désormais transformer (5) en un problème biconvexe. La transformation biconvexe permet de conserver l'avantage de la faible dimension et de la vitesse d'exécution. Ainsi, la relaxation biconvexe (BCR)[1] prend la forme suivante :

$$\begin{aligned} & min_{X,Q_i,i \in \{\mathcal{B} \cup \mathcal{E}\}} \text{ tr}(X^T CX) + \frac{\alpha}{2} \sum_{i \in \{\mathcal{B} \cup \mathcal{E}\}} ||Q_i - L_i X||_F^2 - \frac{\beta}{2} \sum_{j \in \mathcal{E}} ||Q_j||_F^2 \\ & \text{s.c } ||Q_i||_F^2 \leqslant b_i, \forall i \in \{\mathcal{B} \cup \mathcal{E}\} \end{aligned} \tag{6}$$

avec $\alpha > \beta > 0$ les paramètres de relaxation. On a ainsi transformé la contrainte d'égalité $||Q_i||_F^2 = b_i$, $\forall i \in \mathcal{E}$, en contrainte d'inégalité $||Q_i||_F^2 \leqslant b_i$, $\forall i \in \mathcal{E}$, et en ajoutant un terme de pénalité quadratique à la fonction objectif. Cette pénalité force la contrainte d'inégalité sur $\mathcal{E}$ à être parfaitement satisfaite. Les paramètres de relaxation sont choisis en fixant le ratio $\frac{\alpha}{\beta}$ à 2 et en respectant $\beta = ||C||_2$.

3.2 L'algorithme Alternating minimization (AM)

Un point important de (6) est que grâce à la biconvexité, on peut à la fois trouver un optimum global sur X ou sur Q. On utilise l'algorithme AM pour trouver une approximation de cette solution. Ceci se fait en deux étapes :

— D'abord une minimisation par rapport à Q_i. La valeur optimale de Q_i peut être trouvée en minimisant la partie quadratique de la fonction objectif, puis en projetant ce résultat sur une boule unité (associée à lanorme de Frobenius) de rayon $\sqrt{b_i}$. Le minimum de la fonction objectif est atteint en $\frac{\alpha}{\alpha - \beta_i} L_i X$ avec $\beta_i = 0$ si $i \in \mathcal{B}$ et

$\beta_i = \beta$ si $i \in \mathcal{E}$. Ainsi, en projetant sur la boule on actualise Q_i de la manière suivante :

$$Q_i \leftarrow \frac{L_i X}{||L_i X||_F} min\{\sqrt{b_i}, \frac{\alpha}{\alpha - \beta_i}||L_i X||_F\} \tag{7}$$

— Puis une minimisation par rapport à X, où l'on cherche à résoudre un problème de moindre carrés. On cherche à résoudre :

$$X \leftarrow argmin_{X \in \mathbb{R}^{NXr}} tr(X^T C X) + \frac{\alpha}{2} \sum_{i \in \{\mathcal{E} \cup \mathcal{B}\}} ||Q_i - L_i X||_F^2 \tag{8}$$

qui est atteint en :

$$X \leftarrow (C + \alpha \sum_{i \in \{\mathcal{E} \cup \mathcal{B}\}} L_i^T L_i)^{-1} (\sum_{i \in \{\mathcal{E} \cup \mathcal{B}\}} L_i^T Q_i) \tag{9}$$

Algorithme

Algorithme AM dans le cas de la relaxation biconvexe :

1. inputs : C, $\{L_i\}$, b_i, α et β. Output : X
2. Initialisation de X
3. Calcul de : $M = (C + \alpha \sum_{i \in \{\mathcal{E} \cup \mathcal{B}\}} L_i^T L_i)^{-1}$
4. Tant qu'il n'y a pas de convergence :
   $Q_i \leftarrow \frac{L_i X}{||L_i X||_F} min\{\sqrt{b_i}, \frac{\alpha}{\alpha - \beta_i}||L_i X||_F\}$
   $X \leftarrow M(\sum_{i \in \{\mathcal{E} \cup \mathcal{B}\}} L_i^T Q_i)$
   End while

Voyons maintenant comment et pourquoi initialiser X. Comme (6) est biconvexe, on peut minimiser par rapport à $\{Q_i\}$ ou par rapport à X, mais un minimum global de la loi jointe n'est pas garantit. De plus, en utilisant l'algorithme AM, il se peut que l'actualisation de X soit bloqué dans un minimum local, surtout si les variables sont mal initialisées. On va alors utiliser une méthode permettant d'initialiser X de manière assez proche du minimum global du problème de relaxation biconvexe, ce qui va favoriser une convergence rapide de l'algorithme AM.
On cherche donc à trouver une bonne initialisation au problème (4), où X peut être de rang 1 ou plus, et en se concentrant simplement sur des contraintes d'égalité. Etant donné

que C est une matrice symétrique définie positive, alors on peut le réécrire sous la frome $C = U^T U$, et en utilisant un changement de variable $\tilde{X} = UX$, on peut réécrire (1) de la manière suivante :

$$\begin{aligned} & min_{X \in \mathbb{R}^{N \times r}} ||\tilde{X}||_F^2 \\ & \text{s.c } < \tilde{A}_i, \tilde{X}\tilde{X}^T >= b_i, \forall i \in \mathcal{E} \end{aligned} \tag{10}$$

où $\tilde{A}_i = U^T A_i U - 1$. Ainsi, pour initialiser notre algorithme AM, on fait simplement le changement de variable $\tilde{X} = UX$, afin de transformer notre problème BCR, en un problème de la forme (10). Ensuite, on va chercher à initialiser $\tilde{X_0}$. Pour se faire, on calcule $Z = \frac{1}{|\mathcal{E}|} \sum_{i \in \mathcal{E}} b_i L_i^T L_i$, ainsi que la plus grande valeur propre λ et son vecteur propre associé r. On pose alors $\tilde{X_0} = \lambda * r$, et enfin, on détermine l'initialisation du problème en inversant le changement de variable $X_O = U^{-1}\tilde{X_0}$.
L'avantage de la relaxation biconvexe, est qu'elle peut être appliquée sur différents types de problème en programmation SDP. En plus de cela, étant donné que les problèmes BCR sont biconvexe, il y a une garantie de convergence, que ce soit via l'algorithme AM, où d'autres algorithmes de descente de gradient.

4 La méthodes SDCut

La Relaxation Biconvexe est une bonne alternative à la méthode SDCut [2] qui obtenait d'excellents résultats malgré des coûts computationnels long, qui elle même obtient des résultats bien supérieurs au BNCut.

Si on considère le même problème que précédemment (4), on peut le formuler de la façon suivante :

$$\begin{aligned} & min_{X \succcurlyeq 0} \langle X, A \rangle + \sigma(||X||_F^2 - \eta^2) \\ \text{s.c. } & Q_i = L_i X, ||Q_i||_F^2 = b_i, \forall i \in \mathcal{E} \\ & Q_j = L_j X, ||Q_j||_F^2 \leqslant b_j, \forall j \in \mathcal{B} \end{aligned} \tag{11}$$

Et Wang et Al. ont pu simplifier le problème dual (11) de la façon suivante :

$$\begin{aligned} & max_u - \frac{1}{4\sigma} ||C(u)_-||_F^2 - u^T b - \sigma\eta^2 \\ \text{s.c. } & u_j \geqslant 0, \forall j \in \mathcal{B} \end{aligned} \tag{12}$$

Avec $C(u) = \sum_{i=1}^m u_i B_i + A$ et tel que la partie positive d'une matrice X quelconque et négative soient respectivement

$$X_+ = \sum_{\lambda_i > 0} \lambda_i p_i p_i^T, X_- = \sum_{\lambda_i < 0} \lambda_i p_i p_i^T$$

avec p_i les vecteurs propres associés aux valeurs propres.

Après avoir obtenu dans u à l'aide d'un algorithme d'optimisation (Wang et Al ont utilisé Limited-memory BFGS), on peut retrouver X de la façon suivante :

$$X^* = \frac{1}{2\sigma}(C(u^*)_+ - C(u^*)) = -\frac{1}{2\sigma}C(u^*)_- \tag{13}$$

5 Exemple d'utilisation

5.1 Segmentation d'images

Considérons une image contenant N pixels. La segmentation entre le premier plan et l'arrière plan d'une image peut se faire grâce à de la théorie des graphes, où les arrêtes du graphe représentent les similarités entre les différentes paires de pixels. Ce problème peut être modélisé comme un problème NP-complexe :

$$min_{\mathrm{x}\in\{-1,1\}^N}\ \mathrm{x}^T L\mathrm{x} \tag{14}$$

où L représente une matrice de poids (similarité entre les pixels) et x contient le label de la région à laquelle le pixel appartient. On transforme alors ce problème en un problème en plus grande dimension :

$$\begin{aligned} &min_{X\in S^+_{NXN}} tr(L^T X) \\ &\text{s.c } diag(X) = 1, rang(X) = 1 \end{aligned} \tag{15}$$

En retirant la contrainte de rang non convexe, le problème (12) devient un problème de programmation semi-définie positive que l'on peut résoudre grâce à de l'optimisation convexe. Le problème de l'approche SDP est que l'on ne peut pas trouver de solution exacte si la solution du problème est de rang supérieur à 1. Ceci n'est pas le cas pour l'approche BCR qui est parfaitement efficace en terme de temps de calcul pour des problèmes en grandes dimensions, et peut facilement prendre en compte les contraintes de rang.
Considérons un problème basé sur une segmentation SDP, présenté par Stella X. Yu et Jianbo Shi. Ce problème contient trois types de contraintes sur les pixels :

$$\begin{aligned} (t_f^T P\mathrm{x})^2 &\geqslant \kappa||t_f^T P\mathrm{x}||_1^2 \\ (t_b^T P\mathrm{x})^2 &\geqslant \kappa||t_b^T P\mathrm{x}||_1^2 \\ ((t_f - t_b)^T P\mathrm{x})^2 &\geqslant \kappa||(t_f - t_b)^T P\mathrm{x}||_1^2 \end{aligned}$$

Avec $\kappa \in [0, 1]$, $P = D^{-1}W$ représente la matrice de similarité normalisée et t_f et t_b représente des variables indicatrices d'appartenance du pixel au foreground ou au background. Ces contraintes permettent de faire en sorte que la pre-labellisation imposée par

l'utilisateur soit respectée, mais aussi que les pixels proches (dans la matrice de similarité) aient le même label.La matrice de proximité W est donnée par :

$$W_{i,j} = \begin{cases} exp\{-\frac{||f_i - f_j||_2^2}{\gamma_j^2} - \frac{d(i,j)^2}{\gamma_d^2}\} & \text{si } d(i,j) < r \\ 0 & \text{sinon.} \end{cases}$$

où f_i représente l'histogramme de couleur du super-pixel (région de pixels) i et d(i,j) représente la distance entre le pixel i et le pixel j. En considérant ces contraintes et en utilisant le fait que $X = Y^T Y$, on peut réécrire le problème (12), en un problème de la forme (4) :

$$\begin{aligned} & min_{Y \in \mathcal{R}^{NXr}} \text{ tr}(Y^T L Y) \\ \text{s.c } & tr(Y^T A_i Y) = 1, \forall i = 1, ..., N \\ & tr(Y^Y B_2 Y) \geqslant \kappa ||t_f^T P \text{x}||_1^2 \\ & tr(Y^Y B_3 Y) \geqslant \kappa ||t_b^T P \text{x}||_1^2 \\ & tr(Y^Y B_4 Y) \geqslant \kappa ||(t_f - t_b)^T P \text{x}||_1^2 \\ & tr(Y^Y B_1 Y) = 0 \end{aligned} \tag{16}$$

où r est le rang de la solution souhaitée, $B_1 = \mathbf{1}\mathbf{1}^T$, $B_2 = P t_f t_f^T P$, $B_3 = P t_b t_b^T P$, $B_4 = P(t_f - t_b)(t_f - t_b)^T P$, $A_i = e_i e_i^T$ où les e_i sont les éléments de la base canonique de $\mathbb{R}^n$.

S. Shah et al. on comparé l'efficacité de l'approche BCR par rapport à d'autres approches pour résoudre des problèmes BQP, comme SDCut et BNCut. Ainsi, les résultats de l'approche BCR sont aussi pertinents que ceux de l'approche SDCut et meilleurs que ceux de l'approche BNCut (en terme de segmentation du premier plan). BCR est aussi 35 fois plus rapide que SDCut et demande moins de temps de calcul. De plus, l'approche BCR permet de trouver une solution de rang 2 alors que SDCut nécessité une solution de rang 7.

5.2 Co-segmentation

On peut aussi appliquer une approche BCR sur de la co-segmentation, où l'on fait une segmentation du même objet sur différentes images de manière simultanée. La co-segmentation est basée sur un dilemne entre deux critères la pertinence des couleurs et de l'espace au sein d'une même image, et la discrimination entre le premier plan et l'arrière

plan sur différentes images. On peut modéliser le problème de la manière suivante :

$$\begin{aligned} & min_{\mathrm{x}\in\{\pm 1\}^N} \mathrm{x}^T A \mathrm{x} \\ & \text{s.c } (\mathrm{x}^T \delta_i)^2 \leqslant \lambda^2, \forall i = 1, ..., M \end{aligned} \tag{17}$$

où M est le nombre d'images et $N = \sum_{i=1}^{M} N_i$, le nombre de pixels total sur toutes les images. Pour résoudre ce problème par une approche BCR, on doit réécrire (14) sous la forme (4) :

$$\begin{aligned} & min_{X\in\mathbb{R}^{N\times r}} \text{ tr}(X^T A X) \\ \text{s.c } & tr(X^T Z_i X) = 1, \forall i = 1, ..., N \\ & tr(X^T \Delta_i X) \leqslant \lambda^2, \forall i = 1, ..., M \end{aligned} \tag{18}$$

où $\Delta_i = \delta_i \delta_i^T$, $Z_i = e_i e_i^T$ et $A = A_b + \frac{\mu}{N} A_w$ (où A_w est la matrice de proximité intra-image, et A_b est la matrice de coût sur le clustering inter-image).

Les auteurs ont comparé une approche BCR à une approche de factorisation en faible rang (LR), et à l'approche SDCut concernant la co-segmentation. L'approche BCR converge, 9 fois plus vite que l'approche SDCut et 60 fois plus vite que l'approche LR. De plus, le minimum de la fonction objectif trouvé par BCR est bien plus petit que celui des approches LR et SDCut.

6 Implémentation

En utilisant les travaux de Wang et Al. ainsi que de Shah et Al., nous avons tenté de reproduire les résultats obtenus par la méthode SDCut et BCR dans le cadre de la segmentation d'image. Si les résultats de SDCut ont pu être reproduit, nous n'avons pour l'instant toujours pas réussi à implémenter l'algorithme AM, nous n'obtenons pas une convergence satisfaisante.

Nos simulations sont disponibles sur le github suivant.

7 Bibliographie

1. Sohil Shah, Abhay Kumar Yadav : Biconvex Relaxation for Semidefinite Programming in Computer Vision (2016)
2. Wang, P., Shen, C., van den Hengel, A. : A fast semidefinite approach to solving binary quadratic problems (2013)

3. Subhransu Maji, Nisheeth K. Vishnoi, and Jitendra Malik : Biased Normalized Cuts (2011)
4. Vandenberghe, L., Boyd, S. : Semidefinite programming (1996)