Apprentissage de dictionnaire par différentiation automatique pour la résolution de problèmes inverses

Benoît MALÉZIEUX[1, 2], Thomas MOREAU[1], Matthieu KOWALSKI[2]

[1]Université Paris-Saclay, Inria, CEA
91120, Palaiseau, France

[2]L2S, Université Paris-Saclay–CNRS–CentraleSupelec
91190, Gif-sur-Yvette, France

benoit.malezieux@inria.fr, thomas.moreau@inria.fr
matthieu.kowalski@universite-paris-saclay.fr

Résumé – Les problèmes inverses consistent à reconstruire un signal étant donné une observation bruitée. Quand le signal d'intérêt ne peut être observé directement, la résolution doit se faire de manière non supervisée. Dans ce cas, il est nécessaire d'introduire de l'information a priori sur le signal pour pouvoir résoudre le problème. Nous comparons deux méthodes d'apprentissage de dictionnaire parcimonieux par différentiation automatique dans un contexte de problème inverse à l'Analyse et à la Synthèse.

Abstract – Inverse problems consist of recovering a signal given a noisy transformation when no ground truth is available. Some prior knowledge about the signal is required and must be provided to the reconstruction algorithm to solve the problem. In this work, we learn the prior in the unsupervised setting by leveraging the sparsity property of natural signals. We compare two sparse dictionary learning methods based on automatic differentiation and sparse coding algorithms, and derived from Analysis and Synthesis.

1 Motivation

La résolution de problèmes inverses est un enjeu majeur dans une multitude de domaines, de l'astrophysique à l'imagerie médicale. Ils sont généralement mal posés et il est rare que l'on ait accès à une vérité terrain pour les évaluer. Une approche courante de résolution est de poser un problème d'optimisation qui intègre une information a priori sur la structure de la solution. L'efficacité computationnelle et la qualité des solutions dépendent de manière significative de cette information.

Une autre approche consiste à apprendre l'information sur des signaux éventuellement bruitées pour restaurer des données du même type, comme en apprentissage de dictionnaire [1].

Cependant, les méthodes supervisées ne sont pas applicables lorsque l'on ne peut pas observer les signaux sans dégradation et réduction de dimension. Ici, nous travaillons dans un cadre non supervisé et notre but est de comprendre dans quelle mesure il est possible d'apprendre l'information à partir des observations afin de faciliter la reconstruction du signal.

Contributions. Nous proposons d'étendre des résultats récemment obtenus dans [2] au cas des problèmes inverses où les seules données accessibles sont des transformations bruitées des signaux que l'on cherche à reconstruire. Nous étudions l'usage de la différentiation automatique pour apprendre une représentation parcimonieuse et nous comparons deux méthodes d'apprentissage de dictionnaires à l'Analyse et à la Synthèse reposant sur l'optimisation déroulée [3].

2 Problème et état de l'art

Nous considérons le cas où les signaux observés $\mathbf{Y} \in \mathbb{R}^{m \times K}$ sont la somme d'une transformation linéaire $A \in \mathbb{R}^{m \times n}$ de signaux inconnus $\mathbf{X} \in \mathbb{R}^{n \times K}$ et d'un bruit blanc Gaussien $\mathbf{B} \in \mathbb{R}^{m \times K}$:

$$\mathbf{Y} = A\mathbf{X} + \mathbf{B} \ . \tag{1}$$

On cherche à reconstruire $\mathbf{X}$ à partir des observations $\mathbf{Y}$. Dans un contexte non supervisé, il n'est pas possible d'utiliser un jeu d'entraînement où l'on connaît à la fois $\mathbf{X}$ et $\mathbf{Y}$. De plus, quand $m < n$, A n'est pas inversible, et le problème ne peut pas être résolu sans information sur le signal. L'Analyse et la Synthèse [4] permettent de reconstruire $\mathbf{X}$ en tirant parti de cette information ainsi que de la parcimonie des signaux naturels.

A l'Analyse, on utilise une transformation $\Gamma \in \mathcal{C}_A$, où $\mathcal{C}_A$ est un ensemble de contraintes, pour convertir le signal en une représentation parcimonieuse. On pose $\lambda > 0$ le paramètre de régularisation. Le problème d'optimisation à résoudre est le suivant :

$$\min_{\mathbf{X} \in \mathbb{R}^{n \times K}, \Gamma \in \mathcal{C}_A} F_A(\mathbf{X}, \Gamma) \triangleq \frac{1}{2} \left\| A\mathbf{X} - \mathbf{Y} \right\|_2^2 + \lambda \left\| \Gamma^T \mathbf{X} \right\|_1 \ . \tag{2}$$

A la Synthèse, on fait l'hypothèse que le signal est une combinaison linéaire d'un petit nombre d'atomes. Ces atomes sont les colonnes d'un dictionnaire $\Lambda \in \mathcal{C}_S$, où $\mathcal{C}_S$ est un ensemble de contraintes. Le problème d'optimisation à résoudre devient :

Algorithm 1 ISTA

$\mathbf{Y}$, A, Λ, λ, N
$\mathbf{Z_0} = 0$, $n = 0$
Calculer la constante de Lipschitz L of $(A\Lambda)^T(A\Lambda)$
while $n < N$ **do**
  $\mathbf{u_{n+1}} \leftarrow \mathbf{Z_n} - \frac{1}{L}(A\Lambda)^T((A\Lambda)\mathbf{Z_n} - \mathbf{Y})$
  $\mathbf{Z_{n+1}} \leftarrow ST_{\frac{\lambda}{L}}(\mathbf{u_{n+1}})$
  $n \leftarrow n + 1$
end while

Algorithm 2 Condat-Vu

$\mathbf{Y}$, A, Γ, λ, N
(τ, σ) t.q. $\frac{1}{\tau} - \sigma \left\|\Gamma^T\right\|^2 \geq \frac{\|A\|^2}{2}$
$\mathbf{P_0} = 0$, $\mathbf{D_0} = 0$, $n = 0$
while $n < N$ **do**
  $\mathbf{P_{n+1}} \leftarrow \mathbf{P_n} - \tau A^T(A\mathbf{P_n} - Y) - \tau\Gamma\mathbf{D_n}$
  $\mathbf{D_{n+1}} \leftarrow \mathrm{prox}_{\sigma\lambda(\|\cdot\|_1^*)}(\mathbf{D_n} + \sigma\Gamma^T(2\mathbf{P_{n+1}} - \mathbf{P_n}))$
  $n \leftarrow n + 1$
end while

$$\min_{\mathbf{Z}\in\mathbb{R}^{L\times K}, \Lambda\in\mathcal{C}_S} F_S(\mathbf{Z}, \Lambda) \triangleq \frac{1}{2}\|A\Lambda\mathbf{Z} - \mathbf{Y}\|_2^2 + \lambda\|\mathbf{Z}\|_1 \quad . \quad (3)$$

Nous proposons d'apprendre Γ ou Λ en utilisant uniquement les observations dégradées $\mathbf{Y}$, sans accès aux signaux de départ $\mathbf{X}$, à la différence de travaux antérieurs [5, 6]. Notre méthode repose sur la formulation d'un problème d'optimisation à deux niveaux et sur sa résolution avec la différentiation automatique.

3 Optimisation bi-niveau pour l'apprentissage de dictionnaires

Les modèles d'apprentissage de dictionnnaires à l'Analyse et à la Synthèse peuvent être réécrits comme des problèmes d'optimisation à deux niveaux pour minimiser la fonction de coût uniquement par rapport à Λ ou Γ :

$$\min_{\Lambda\in\mathcal{C}_S} F_S(\mathbf{Z}^*(\Lambda), \Lambda) \quad \text{t.q.} \quad \mathbf{Z}^*(\Lambda) = \operatorname*{argmin}_{\mathbf{Z}\in\mathbb{R}^L} F_S(\mathbf{Z}, \Lambda) \ , \quad (4)$$

$$\min_{\Gamma\in\mathcal{C}_A} F_A(\mathbf{X}^*(\Gamma), \Gamma) \quad \text{t.q.} \quad \mathbf{X}^*(\Gamma) = \operatorname*{argmin}_{\mathbf{X}\in\mathbb{R}^n} F_A(\mathbf{X}, \Gamma) \ . \quad (5)$$

Pour optimiser F_S (resp. F_A) dans (4) (resp. (5)), il faut calculer le gradient (ou un sous-gradient) par rapport à Λ (resp. Γ). En prenant l'exemple de la Synthèse, on peut définir la fonction de coût par rapport à Λ :

$$F(\Lambda) \triangleq F_S(Z^*(\Lambda), \Lambda) \ . \quad (6)$$

Une fois $Z^*(\Lambda)$ connu, le théorème de Danskin (1967) montre que le gradient est égal à $\nabla_2 F_S(Z^*(\Lambda), \Lambda)$, où ∇_2 désigne la différentiation par rapport à la deuxième variable, et il est possible d'optimiser la fonction de coût par descente de gradient [1]. La principale limite est que connaître la valeur de $Z^*(\Lambda)$ nécessite un nombre important d'itérations, et il est difficile de procéder ainsi sur un grand jeu de données.

3.1 Différentiation automatique

Une autre solution pour calculer le gradient de F_S (ou de F_A) est d'utiliser la différentiation automatique, comme proposé dans [3]. L'idée consiste à construire un réseau de neurones à partir de l'agorithme d'optimisation utilisé pour calculer $\mathbf{Z}^*(\Lambda)$ ou $\mathbf{X}^*(\Gamma)$. La sortie du réseau est une approximation

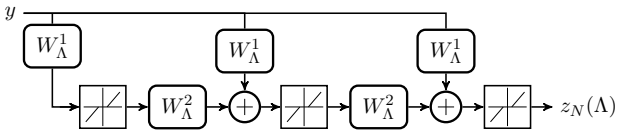

FIGURE 1 – LISTA

de la solution du problème, et le réseau lui-même est paramétré par le dictionnaire. Le gradient est ensuite calculé par différentiation automatique puis utilisé pour trouver un dictionnaire.

Synthèse. On cherche à approcher la solution $\mathbf{Z}^*(\Lambda)$ du problème du LASSO [7]. On peut dérouler un nombre N d'itérations de ISTA [8] ou sa version accélérée FISTA [9] pour obtenir une approximation $\mathbf{Z_N}(\Lambda)$ de $\mathbf{Z}^*(\Lambda)$. L'algorithme 1 rappelle ISTA. En notant $W_\Lambda^1 = \frac{1}{L}(A\Lambda)^T$ et $W_\Lambda^2 = (I - \frac{1}{L}(A\Lambda)^T A\Lambda)$, le réseau de neurones est construit sur le modèle de LISTA [3] illustré dans Figure 1. Dans [2], nous étudions la Jacobienne de $Z_N(\Lambda)$ pour montrer que dérouler ISTA à la Synthèse conduit à des instabilités numériques dans l'estimation du gradient du dictionnaire dans le cas où le nombre d'itérations est trop important. En effet, les erreurs dues à la mauvaise estimation du support s'accumulent au fil des itérations et impactent négativement l'estimation de la jacobienne. La stabilité n'est garantie que lorsque le support du code parcimonieux est bien estimé. De plus, nous montrons empiriquement que l'estimation du dictionnaire reste efficace quand le nombre d'itérations est faible. Ces résultats s'appliquent dans un contexte de problème inverse, d'après Proposition 3.1. Par la suite, nous n'utilisons donc qu'un faible nombre de couches (moins de 30) pour limiter les instabilités.

Proposition 3.1 *Soit Λ_l la ligne l de Λ. A l'itération N de ISTA, la Jacobienne de $z_{N+1}(A\Lambda)$ par rapport à Λ_l peut être écrite en fonction de $(J^i_{N+1}(A\Lambda))_{1\leq i\leq m}$ où $J^i_{N+1}(A\Lambda)$ est la Jacobienne de la ligne i de $A\Lambda$ calculée dans [2] :*

$$\frac{\partial(z_{N+1}(A\Lambda))}{\partial\Lambda_l} = \sum_{i=1}^{m} A_{i,l} J^i_{N+1}(A\Lambda)$$

La Jacobienne de $z_{N+1}(A\Lambda)$ converge à la même vitesse que les $(J^i_{N+1}(A\Lambda))_{1\leq i\leq m}$.

Analyse. On cherche à approcher la solution $\mathbf{X}^*(\Gamma)$ du problème à l'Analyse. On peut dérouler N itérations de l'algorithme primal-dual de Condat-Vu [10, 11] pour obtenir une

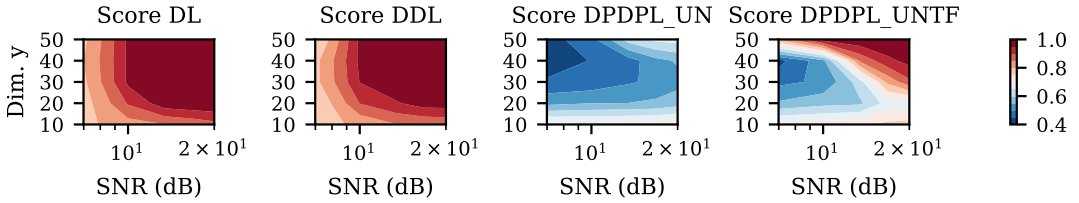

FIGURE 2 – Performance en fonction du SNR et de la dimension de l'observation pour un dictionnaire de 50 atomes.

approximation $\mathbf{X_N}(\Gamma)$ de $\mathbf{X}^*(\Gamma)$. Un réseau de neurones est construit à partir de cet algorithme sur le même principe qu'à la Synthèse. L'algorithme 2 rappelle l'approche Condat-Vu. Les paramètres sont le dictionnaire Γ, et les pas τ et σ.

3.2 Contraintes et optimisation.

En apprentissage de dictionnaire, il est fréquent de borner la norme des atomes afin d'éviter les problèmes d'invariance d'échelle. On note **UN** l'ensemble $\{U \text{ t.q. } \|U_i\|_2 \leq 1 \; \forall i \in I\}$ où I est l'ensemble des indices des colonnes de U. Nous étudions aussi le cas où Λ et Γ sont formés de filtres de convolutions, noté **UN + Conv**. L'optimisation se fait à l'aide d'une descente de gradient projetée avec une recherche linéaire. La projection correspond ici à une normalisation des atomes.

Dans [12], les auteurs remarquent que la contrainte **UN** n'est pas suffisante pour apprendre un dictionnaire à l'Analyse. Ils proposent la contrainte suivante, notée **UNTF** : $\{U \text{ t.q. } UU^T = I, \forall i \in I \, \|U_i\|_2 = 1\}$, sous ensemble de la variété de Stiefel. On optimise sur ce sous-ensemble en calculant le nouveau point sur la variété à l'aide de la transformée de Cayley et d'une recherche linéaire, comme proposé dans [13] et [14].

Amélioration de la précision La méthode présentée est efficace quand elle est utilisée avec un faible nombre d'itérations en raison du coup de calcul lié à la rétropropagation et des instabilités numériques. Il est possible d'accélérer la convergence des algorithmes en apprenant les pas de descente dans (F)ISTA [15] ou Condat-Vu, pour améliorer la précision des résultats à la sortie du réseau. Nous proposons un apprentissage en deux étapes pour limiter le degré de liberté des paramètres, ce qui rend la convergence plus stable d'après nos expérimentations.

1. Apprentissage du dictionnaire avec un pas fixé, correspondant à une condition de convergence de l'algorithme d'optimisation.
2. Apprentissage des pas pour accélérer la convergence et gagner en précision sur $\mathbf{Z}_N(\Lambda)$ et $\mathbf{X}_N(\Gamma)$, tout en poursuivant l'apprentissage du dictionnaire qui est maintenant bien initialisé.

4 Expériences et résultats numériques

Dans [2], nous montrons que dérouler un nombre restreint d'itérations d'un algorithme d'optimisation permet de retrouver le dictionnaire en réduisant le temps de calcul, surtout pour des jeux de données de grande taille, en prenant l'exemple de signaux cérébraux multivariés. Dans cette partie, nous étendons ces résultats en comparant DDL (Deep Dictionary Learning) à la Synthèse [5] et DPDPL (Deep Primal Dual Prior Learning) à l'Analyse [6] sur deux exemples non supervisés : le débruitage sur données synthétiques et l'inpainting sur une image réelle.

4.1 Débruitage sur données synthétiques

Nous vérifions ici la capacité des algorithmes à retrouver un dictionnaire **UNTF** aléatoire de 50 atomes utilisé pour générer les données dans un contexte de débruitage. A la Synthèse, on tire aléatoirement les codes selon une loi Bernouilli-Gaussienne. A l'Analyse, on applique la méthode de génération des données proposée dans [4]. Le score utilisé pour juger de la qualité du prior appris, indépendamment du signe et de l'ordre des atomes, est le suivant :

$$S(\Lambda, \Lambda_{ref}) = \max_{\sigma \in \mathfrak{S}_n} \frac{1}{n} \sum_{i=1}^{n} |(\Lambda)^T_{\sigma(i)} (\Lambda_{ref})_i| \; . \qquad (7)$$

Il s'agit de la moyenne des similarités cosinus, en valeur absolue, du meilleur appariemment d'atomes entre les deux dictionnaires. Plus le score est proche de 1, mieux le dictionnaire est retrouvé. Nous comparons d'abord le score maximal sur plusieurs valeurs de λ et du nombre d'itérations (compris entre 5 et 50) en fonction du rapport signal à bruit (SNR) en dB et défini par $10 \log_{10}(\frac{\sigma^2}{\sigma_b^2})$ où σ^2 est la variance du signal et σ_b^2 est la variance du bruit, et en fonction de la dimension des observations Y. Les résultats sont rapportés dans Figure 2. DDL se comporte comme un algorithme d'apprentissage de dictionnaire (DL) standard. L'expérience confirme que la contrainte **UN** n'est pas suffisante à l'Analyse. La contrainte **UNTF** ne fonctionne que quand la dimension du signal est proche du nombre d'atomes et quand le SNR est grand.

4.2 Exemple de l'inpainting

Dans un contexte de problème inverse, la dimension des mesures m est souvent plus petite que la dimension du signal n, et a fortiori de la dimension des codes parcimonieux L. Cette réduction de dimension implique qu'une partie de l'information contenue dans le signal est perdue. Lorsque la matrice de mesure est unique, on peut seulement espérer identifier un dictionnaire sur un sous-espace de dimension m du signal. Pour pouvoir apprendre un dictionnaire plus informatif, il est nécessaire d'avoir accès à plusieurs matrices de mesure, de telle sorte

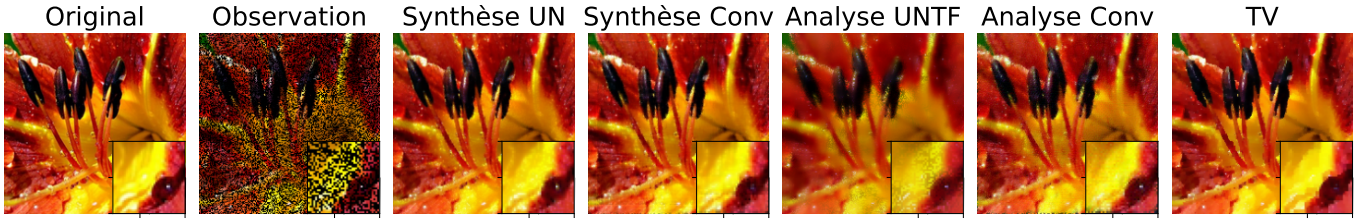

FIGURE 3 – Inpainting sur une image 200 × 200 avec 50% de pixels manquant. La Synthèse est plus performante que l'Analyse. PSNR : Synthèse UN 33.4dB ; Synthèse Conv 33.3dB ; Analyse UNTF 29.6dB ; Analyse Conv 30.8dB ; TV 32.7dB.

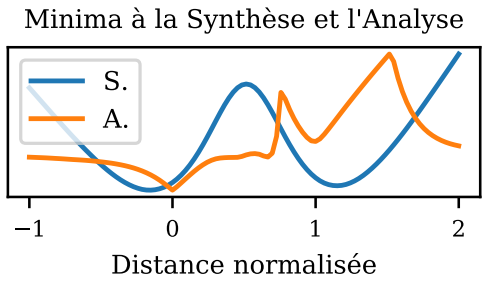

FIGURE 4 – Comparaison des fonctions de coût à l'Analyse et à la Synthèse entre deux minima locaux.

que l'intersection des noyaux de ces matrices soit le singleton nul. Cette condition est vérifiée dans plusieurs types de problèmes inverses, par exemple en inpainting où le dictionnaire est appris sur des patches de petite taille. La matrice de mesure, ici un masque binaire, est unique pour l'image entière mais pas pour les patches.

Le traitement se fait sur des patches de petite taille dans le cas des contraintes **UN** et **UNTF**, et directement sur l'image dans le cas des convolutions. Les images sont présentées Figure 3. La qualité des images reconstruites est meilleure à la Synthèse. Cependant, les convolutions améliorent les performances à l'Analyse. Afin de mieux comprendre ces différences de résultats et l'impact de l'initialisation sur le dictionnaire final, nous avons représenté les fonctions de coûts entre plusieurs minima locaux dans Figure 4. L'Analyse semble moins bien conditionnée que la Synthèse dont la fonction de coût est beaucoup plus lisse, et nos expériences montrent que la première n'est pas robuste à l'initialisation aléatoire contrairement à la deuxième.

5 Conclusion

Nous avons comparé deux méthodes de résolution de problèmes inverses, l'une à partir de la formulation à la Synthèse, et l'autre à partir de la formulation à l'Analyse. Les réseaux de neurones utilisés sont construits à partir d'algorithmes d'optimisation. Les résultats expérimentaux montrent que la Synthèse parvient plus facilement à reconstruire le signal alors que l'Analyse nécessite des contraintes plus strictes pour fonctionner. L'intérêt de ces méthodes est leur interprétabilité et leur faible nombre de paramètres, comparé à des réseaux de neurones standards. De plus, elles peuvent être appliquées dans un contexte non supervisé.

Références

[1] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696, 2009.

[2] Benoît Malézieux, Thomas Moreau, and Matthieu Kowalski. Understanding approximate and unrolled dictionary learning for pattern recovery. *International Conference on Learning Representations*, 2022.

[3] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. *International conference on machine learning*, pages 399–406, 2010.

[4] Michael Elad, Peyman Milanfar, and Ron Rubinstein. Analysis versus synthesis in signal priors. *Inverse Problems*, 23 :947, 2007.

[5] Meyer Scetbon, Michael Elad, and Peyman Milanfar. Deep k-svd denoising. *IEEE Transactions on Image Processing*, 30 :5944–5955, 2021.

[6] Mingyuan Jiu and Nelly Pustelnik. A deep primal-dual proximal network for image restoration. *IEEE Journal of Selected Topics in Signal Processing*, 15(2) :190–203, 2021.

[7] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B*, 58 :267–288, 1996.

[8] Ingrid Daubechies, Michel Defrise, and Christine Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constrains. *Communications on Pure and Applied Mathematics*, 57, 2004.

[9] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, 2 : 183–202, 2009.

[10] Laurent Condat. A primal–dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms. *Journal of Optimization Theory and Applications*, 158(2) :460–479, 2013.

[11] Bằng Công Vũ. A splitting algorithm for dual monotone inclusions involving cocoercive operators. *Advances in Computational Mathematics*, 38(3) :667–681, 2013.

[12] Mehrdad Yaghoobi, Sangnam Nam, Rémi Gribonval, and Mike E Davies. Constrained overcomplete analysis operator learning for cosparse signal modelling. *IEEE Transactions on Signal Processing*, 61(9) :2341–2355, 2013.

[13] Yujie Li, Shuxue Ding, Zhenni Li, Xiang Li, and Benying Tan. Dictionary learning in the analysis sparse representation with optimization on stiefel manifold. In *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1270–1274. IEEE, 2017.

[14] Zaiwen Wen and Wotao Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2) :397–434, 2013.

[15] Pierre Ablin, Thomas Moreau, Mathurin Massias, and Alexandre Gramfort. Learning step sizes for unfolded sparse coding. In *Advances in Neural Information Processing Systems*, pages 13100–13110, 2019.