
Online Dictionary Learning

- Représentation de données par Sparse Coding -

Rapport de projet PRR
Thibault CORDIER

ENSIE
Ecole Nationale Supérieure de l'Informatique pour l'Industrie et l'Entreprise

Contents

1	Introduction au machine learning	1
1.1	Définition	1
1.2	Modes d'apprentissage	2
1.3	Problématiques d'apprentissage	3
2	Feature learning	4
2.1	Définition et Motivations	4
2.2	Dictionary learning	5
3	Sparse Coding	7
3.1	Introduction	7
3.2	Applications	7
3.3	Énoncé du problème	8
4	Online Dictionary Learning	12
4.1	Algorithme	12
4.2	Sparse coding problem	14
4.3	Dictionary update problem	15
5	Validation expérimentale	18

5.1	Base d'apprentissage MNIST	18
5.2	Initialisation du dictionnaire	19
5.3	Construction du modèle	20
5.4	Visualisation des résultats	21
5.5	Analyse des résultats	22
5.6	Limites de la méthode	23
6	Conclusion	24

Chapter 1

Introduction au machine learning

1.1 Définition

L'apprentissage automatique ou statistique - aussi appelé *machine learning* en anglais - est un champ d'étude de l'intelligence artificielle dont l'enjeu est de permettre aux machines (au sens large) d'**apprendre à réaliser une tâche** - c'est à dire être capable d'améliorer leur performance pour réaliser une tâche spécifique - **à partir de données et au moyen de techniques statistiques, sans y être explicitement programmées**. Elle englobe la conception, l'analyse, le développement et l'implémentation de méthodes permettant à ces machines d'évoluer par un processus systématique, et ainsi de remplir des tâches difficiles ou problématiques par des moyens algorithmiques plus classiques.

Le terme *machine learning* a été inventé en 1959 par Arthur Samuel. Développée à partir de l'étude de la reconnaissance des formes et de la théorie de l'apprentissage artificiel, l'apprentissage automatique explore l'étude et la construction d'**algorithmes capables de tirer des enseignements et de faire des prédictions sur les données en construisant un modèle à partir d'intrants**. L'apprentissage automatique est utilisé dans une gamme de tâches de calcul où la conception et la programmation d'algorithmes explicites avec de bonnes performances est difficile ou irréalisable.

Les méthodes de *machine learning* trouvent leurs applications dans les domaines de la bio-informatique, de l'aide aux diagnostics, de l'analyse financière, de la vision par ordinateur, de l'exploration de données, du traitement automatique du langage naturel, de la reconnaissance de formes (visages, écriture), des systèmes de recommandation, des moteurs de recherche, de l'analyse prédictive en matière juridique et judiciaire etc.

1.2 Modes d'apprentissage

Les algorithmes d'apprentissage peuvent se catégoriser selon le mode d'apprentissage qu'ils emploient.

Apprentissage supervisé - Supervised learning

Une base de données d'apprentissage (ou ensemble d'apprentissage) est un ensemble de couples entrée-sortie $(x_n, y_n)_{1 \leq n \leq N}$ avec $x_n \in X$ et $y_n \in Y$, que l'on considère être tirées selon une loi sur $X \times Y$ fixe et inconnue, et tel que $y_n = f(x_n) + w_n$ où w_n est un bruit centré.

La méthode d'apprentissage supervisé utilise cette base d'apprentissage pour déterminer une représentation compacte de f notée g et appelée indistinctement fonction de prédiction, hypothèse ou modèle qui, à une nouvelle entrée x , associe une sortie $g(x)$.

Le but d'un algorithme d'apprentissage supervisé est donc de généraliser pour des entrées inconnues ce qu'il a pu "apprendre" grâce aux données déjà traitées par des experts, ceci de façon "raisonnable". On dit que la fonction de prédiction apprise doit avoir de bonnes garanties en généralisation.

Apprentissage non supervisé - Unsupervised learning

La base de données d'apprentissage peut aussi être un ensemble de données non étiquetées $(x_n)_{1 \leq n \leq N}$ avec $x_n \in X$, que l'on considère être tirées selon une loi sur X fixe et inconnue. Cette base peut néanmoins contenir des structures sous-jacentes que nous pouvons chercher à découvrir. Il peut être possible aussi dans certains cas de catégoriser les données en k classes : $(x_n, \hat{y}_n)_{1 \leq n \leq N}$ avec $x_n \in X$ et $\hat{y}_n \in \hat{Y} = [1, k]$.

La méthode d'apprentissage non supervisé utilise cette base d'apprentissage pour déterminer un modèle g permettant de découvrir ces structures ou, si possible, d'extraire des classes d'individus présentant des caractéristiques communes, qui à une entrée x de X , associe une sortie "classe" $g(x) \in [1, k]$.

Le but d'un algorithme d'apprentissage non supervisé est donc de bien catégoriser des entrées connues de telle sorte de faire ressortir la structure interne de la base d'apprentissage. La qualité d'une méthode de classification est mesurée par sa capacité à découvrir certains ou tous les motifs cachés.

1.3 Problématiques d'apprentissage

Les algorithmes d'apprentissage peuvent aussi se catégoriser selon le problème de prédiction qu'ils cherchent à résoudre.

La **classification** est un problème de catégorisation où l'on cherche à attribuer une catégorie ou *classe* à une nouvelle observation, sur la base d'un ensemble de données d'apprentissage contenant des observations dont l'appartenance à une catégorie est déjà connue.

La **régression** est un problème de prédiction où l'on cherche à estimer les relations entre les variables d'un ensemble de données. Ce problème comprend de nombreuses techniques de modélisation et d'analyse multi-variée. L'accent est mis sur la relation entre une variable expliquée et une ou plusieurs variables explicatives.

Le **clustering** est un problème de regroupement où l'on cherche justement à regrouper un ensemble d'objets de telle sorte que les objets d'un même groupe (appelé *cluster*) soient plus proches - dans un certain sens - les uns des autres que ceux des autres groupes.

L'**estimation de densité** est un problème où l'on cherche à construire une estimation, basée sur un ensemble d'apprentissage, d'une fonction de densité de probabilité sous-jacente non observable. La fonction de densité inobservable est considérée comme la densité selon laquelle une grande population est distribuée. Les données sont généralement considérées comme un échantillon aléatoire de cette population.

La **réduction de la dimensionnalité** ou la réduction des dimensions est un problème où l'on cherche à réduire le nombre de variables considérées dans un ensemble de données en obtenant un ensemble de variables significatives - c'est à dire un ensemble réduit de variables permettant toujours de bien représenter les données.

Chapter 2

Feature learning

2.1 Définition et Motivations

En machine learning, le *feature learning* ou *representation learning* est un ensemble de techniques permettant à une machine de découvrir les représentations nécessaires à la détection ou à la classification de caractéristiques - ou *features* - à partir de données d'entraînement.

Le *feature learning* est motivé par le fait que les tâches d'apprentissage automatique, telles que la classification, nécessitent souvent une saisie mathématique et computationnelle. Cependant, les données du monde réel telles que les images, la vidéo et les données de capteur n'ont pas cédé à des tentatives de définition algorithmique de caractéristiques spécifiques. Une alternative consiste à découvrir de telles caractéristiques ou représentations au moyen d'un examen, sans recourir à des algorithmes explicites.

Supervised vs. Unsupervised feature learning

Supervised feature learning est l'apprentissage de *features* à partir de données étiquetées. L'étiquetage de données permet au système de calculer un terme d'erreur, le degré auquel le système ne parvient pas à produire l'étiquette, qui peut ensuite être utilisé comme rétroaction pour corriger le processus d'apprentissage (réduire l'erreur).

Unsupervised feature learning est l'apprentissage de *features* à partir de données non étiquetées. L'objectif ici est souvent de découvrir des caractéristiques de faible dimension qui capturent une structure sous-jacente aux données d'entrée de grande dimension. Lorsque l'apprentissage est effectué de manière non supervisée, il permet une forme d'apprentissage semi-supervisé où les caractéristiques apprises à partir d'un ensemble de

données non-étiquetées sont ensuite utilisées pour améliorer les performances dans un environnement supervisé avec des données étiquetées.

2.2 Dictionary learning

Le *dictionary learning* développe un ensemble (appelé dictionnaire) d'éléments représentatifs à partir des données d'entrée, de sorte que chaque donnée peut être représentée comme une somme pondérée des éléments représentatifs. Les éléments du dictionnaire et les poids peuvent être trouvés en minimisant l'erreur de représentation moyenne (sur les données d'entrée).

Unsupervised vs. Supervised dictionary learning

Unsupervised dictionary learning n'utilise pas d'étiquettes de données et exploite la structure sous-jacente des données pour optimiser les éléments du dictionnaire. Un exemple d'apprentissage de dictionnaire non supervisé est le codage parcimonieux - ou *sparse coding* - qui vise à apprendre des fonctions de base (éléments de dictionnaire) pour la représentation de données à partir de données d'entrée non étiquetées. Le *sparse coding* peut être appliqué pour apprendre des dictionnaires *over-complete*, où le nombre d'éléments du dictionnaire est plus grand que la dimension des données d'entrée.

Supervised dictionary learning exploite à la fois la structure sous-jacente des données d'entrée et les étiquettes pour optimiser les éléments du dictionnaire. Par exemple, une technique d'apprentissage supervisé de dictionnaire met en application ce type d'apprentissage à des problèmes de classification en optimisant conjointement les éléments du dictionnaire, les poids pour représenter les points de données et les paramètres du classifieur en fonction des données d'entrée. En particulier, un problème de minimisation est formulé selon une fonction objectif fondée sur l'erreur de classification, l'erreur de représentation, une régularisation l_1 sur les poids représentatifs de chaque donnée (pour permettre une représentation éparsée des données) et une régularisation l_2 sur les paramètres du classifieur.

Undercompleted vs. Overcompleted dictionary

Le dictionnaire défini précédemment peut être *undercomplete* si $n < d$ ou *overcomplete* si $n > d$, ce dernier étant une hypothèse typique pour un problème de *sparse dictionary learning*. Le cas d'un dictionnaire *complete* ne fournit aucune amélioration d'un point de vue représentationnel et n'est donc pas considéré.

Undercomplete dictionaries représentent la configuration dans laquelle les données d'entrée réelles se trouvent dans un espace de dimension inférieure. Ce cas est fortement lié à la réduction de la dimensionnalité et à des techniques telles que l'analyse en composantes principales (méthodes PCA) qui nécessitent que les *features* soient orthogonaux. Le choix de ces sous-espaces est crucial pour une réduction efficace de la dimensionnalité, mais ce choix n'est pas trivial. Et la réduction de la dimensionnalité basée sur la représentation du dictionnaire peut être étendue à des tâches spécifiques telles que l'analyse ou la classification des données. Cependant, le principal inconvénient de cette considération est de limiter le choix des *features*.

Overcomplete dictionaries, cependant, n'exigent pas que les *features* soient orthogonaux (ils ne seront jamais une base de toute façon), ce qui permet de rendre les dictionnaires plus flexibles et des représentations de données plus riches.

Chapter 3

Sparse Coding

Nous présentons dans ce chapitre le concept de **sparse coding** à travers le problème de *dictionary learning*, ses applications, ainsi que l'énoncé du problème associé que nous souhaitons résoudre.

3.1 Introduction

Le **sparse coding** - aussi appelée *sparse representation* — est un modèle de représentation qui consiste à modéliser des données sous la forme d'une combinaison linéaire parcimonieuse d'éléments de bases eux-mêmes parcimonieux. Ces éléments sont appelés atomes (*atoms*) et composent ensemble un dictionnaire (*dictionary*).

La parcimonie - ou *sparsity* - d'un vecteur de données se caractérise par la présence de nombreux coefficients nuls qui le composent.

3.2 Applications

Cette modélisation a été largement utilisée dans le traitement du signal, le traitement d'images, l'apprentissage automatique, l'imagerie médicale, le traitement matriciel, ainsi que l'exploration de données et plus encore.

Dans la plupart de ces applications, le signal d'intérêt inconnu est modélisé comme une combinaison parcimonieuse de quelques atomes d'un dictionnaire donné, et ceci est utilisé comme régularisation du problème. Ces problèmes sont typiquement accompagnés

d'un mécanisme d'apprentissage de dictionnaire qui vise à adapter le dictionnaire pour que le modèle corresponde le mieux aux données présentées. L'utilisation de modèles inspirés de méthodes parcimonieuses a conduit à des résultats de pointe dans un large éventail d'applications.

La décomposition linéaire d'un signal utilisant quelques atomes d'un dictionnaire entraîné au lieu d'un dictionnaire prédéfini basé sur des ondelettes par exemple, a récemment conduit à des résultats de pointe pour de nombreux traitements d'images de bas niveau - des tâches telles que le débruitage - ainsi que des tâches de niveau supérieur - telles que la classification - montrant que les modèles parcimonieux sont bien adaptés aux signaux naturels.

Contrairement aux décompositions basées sur l'analyse en composantes principales (ou méthodes PCA) et ses variantes, ces modèles n'imposent pas que les vecteurs de base soient orthogonaux, permettant plus de flexibilité pour adapter la représentation aux données.

Alors que l'apprentissage du dictionnaire s'est avéré crucial pour atteindre des résultats de pointe, résoudre efficacement le problème d'optimisation correspondant est un défi de calcul important, en particulier dans le contexte des ensembles de données à grande échelle impliqués dans l'image qui peuvent inclure des millions d'échantillons d'entraînement.

3.3 Énoncé du problème

Soit \mathbf{X} un ensemble de données (*dataset*) composées de n observations x_i (*features*) exprimées par p paramètres (*parameters*).

$$\mathbf{X} = [x_1, \dots, x_n] \in \mathbb{R}^{p \times n} - \forall i \in [1, n], x_i \in \mathbb{R}^p \quad (3.1)$$

Soit \mathbf{D} un dictionnaire (*dictionary*) que nous souhaitons construire, composé de k vecteurs de base (*atoms*) exprimés aussi par p paramètres, permettant de représenter \mathbf{X} .

$$\mathbf{D} = [d_1, \dots, d_k] \in \mathbb{R}^{p \times k} - \forall i \in [1, k], d_i \in \mathbb{R}^p \quad (3.2)$$

Soit alors \mathbf{R} le *sparse coding* de \mathbf{X} , c'est à dire les coefficients de la *sparse decomposition* optimale r_i des n *features* x_i de \mathbf{X} à travers les k *atoms* de \mathbf{D} .

$$\mathbf{R} = [r_1, \dots, r_n] \in \mathbb{R}^{k \times n} - \forall i \in [1, n], r_i \in \mathbb{R}^k \quad (3.3)$$

Usuellement, le nombre de *features* n est relativement grand comparé à la dimension des *features* p ($n \gg p$) et au nombre d'*atoms* k ($n \gg k$). De plus, l'*under-complete* ($p > k$) tout comme l'*over-complete dictionary* ($p < k$) sont autorisés, selon l'objectif visé.

Les techniques classiques de *dictionary learning* pour construire le dictionnaire \mathbf{D} cherche à optimiser la fonction de coût empirique (*empirical cost function*) définie par :

$$f_n(\mathbf{D}) = \frac{1}{n} \sum_{i=1}^n l(x_i, \mathbf{D}) \quad (3.4)$$

ou l est la fonction de perte (*loss function*) telle que $l(\mathbf{x}, \mathbf{D})$ soit petit si \mathbf{D} est "bon" pour représenter \mathbf{x} .

Généralement, la *sparse coding loss function* $l(\mathbf{x}, \mathbf{D})$ est définie comme la valeur optimale du problème *S-sparse coding* :

$$l(x, \mathbf{D}) = \min_{r \in \mathbb{R}^k} \frac{1}{2} \|x - \mathbf{D}r\|_2^2 + \lambda S(r) \quad (3.5)$$

où λ est le paramètre de régularisation et où $S(\cdot)$ est la *sparsity cost function*.

Le premier terme du *sparse coding objective* (3.5) peut être interprété comme le terme de reconstruction - qui force le modèle à fournir une bonne représentation $\mathbf{D}r$ de x - alors que le second terme peut être interprété comme la pénalité parcimonieuse - qui force la représentation de x à être parcimonieuse. Le paramètre de régularisation λ permet de déterminer l'importance relative des deux termes.

Il est bien connu dans la communauté que la *l_1 -penalty* ($l_1(x) = \|x\|_1$) fournit une solution parcimonieuse, avec peu de coefficients non nuls dans r , bien qu'il n'y ait pas de lien analytique explicite entre la valeur de λ et la parcimonie efficace que produit le modèle.

La *l_0 -penalty* ($l_0(x) = \#\{|x^{(i)}| > 0, i \in [1, p]\}$) est, quant à elle, une mesure plus directe de la parcimonie. Cependant, le *sparse coding problem* avec la *l_0 -penalty* est un problème non convexe - contrairement avec la *l_1 -penalty* qui, en pratique, est plus stable - et il se trouve que résoudre ce problème est NP-difficile.

Nous décidons alors de définir la *sparse coding loss function* $l(\mathbf{x}, \mathbf{D})$ (eq : 3.5) comme la valeur optimale du problème *l_1 -sparse coding* :

$$l(x, \mathbf{D}) = \min_{r \in \mathbb{R}^k} \frac{1}{2} \|x - \mathbf{D}r\|_2^2 + \lambda \|r\|_1 \quad (3.6)$$

où λ est le paramètre de régularisation. Ce problème est aussi connu sous le nom de *basis pursuit* ou le *Lasso*.

Pour empêcher les *atoms* du dictionnaire \mathbf{D} de devenir arbitrairement trop large - ce qui conduirait à rendre les valeurs de r trop étroites - il est aussi nécessaire de contraindre les *atoms* avec la l_2 -norm.

Soit alors \mathcal{C} l'ensemble convexe des dictionnaires vérifiant cette contrainte :

$$\mathcal{C} = \{\mathbf{D} \in \mathbb{R}^{p \times k} \text{ s.c. } \forall i \in [1, \dots, k], \|d_i\|_2 \leq 1\} \quad (3.7)$$

Le *dictionary* \mathbf{D} que nous souhaitons construire (eq 3.2) est désormais sous contrainte.

$$\mathbf{D} = [d_1, \dots, d_k] \in \mathcal{C} - \forall i \in [1, k], d_i \in \mathbb{R}^p \quad (3.8)$$

Nous devons toutefois remarquer que le problème de minimisation de l'*empirical cost function* $f_n(\mathbf{D})$ (eq : 3.4) n'est pas convexe par rapport à \mathbf{D} . Cependant, nous pouvons le réécrire comme un problème d'optimisation par rapport au *dictionary* \mathbf{D} et par rapport aux coefficients de la *sparse decomposition* $\mathbf{R} = [r_1, \dots, r_n]$:

$$f_n(\mathbf{D}, \mathbf{R}) = \|\mathbf{X} - \mathbf{DR}\|_F^2 = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} \|x_i - \mathbf{D}r_i\|_2^2 + \lambda \|r_i\|_1 \quad (3.9)$$

Le problème n'est pas convexe lorsque les deux variables \mathbf{D} et \mathbf{R} sont prises conjointement, mais l'est par rapport à chacune des deux variables \mathbf{D} et \mathbf{R} toutes choses égales par ailleurs.

Nous considérons donc le problème d'optimisation général suivant :

$$\boxed{\min_{\mathbf{D} \in \mathcal{C}, \mathbf{R} \in \mathbb{R}^{k \times n}} \|\mathbf{X} - \mathbf{DR}\|_F^2 = \min_{\mathbf{D} \in \mathcal{C}, \mathbf{R} \in \mathbb{R}^{k \times n}} \frac{1}{n} \sum_{i=1}^n \frac{1}{2} \|x_i - \mathbf{D}r_i\|_2^2 + \lambda \|r_i\|_1} \quad (3.10)$$

Un approche naturelle pour résoudre ce problème est de considérer deux sous problèmes que l'on cherchera à résoudre alternativement.

Le premier problème d'optimisation - *sparse approximation* ou *sparse coding problem* - consiste à trouver un *sparse coding* optimal \mathbf{R} de \mathbf{X} étant donné un *dictionary* \mathbf{D} .

Le deuxième problème d'optimisation - *dictionary update problem* - consiste à mettre à jour le *dictionary* \mathbf{D} étant donné un *sparse coding* \mathbf{R} de \mathbf{X} .

Nous verrons dans la prochaine partie comment nous avons résolu ces deux problèmes d'optimisation.

Enfin, il faut comprendre que nous ne chercherons pas à minimiser parfaitement l'*empirical cost function* $f_n(\mathbf{D})$ (eq 3.4) car il est plus intéressant à chercher à minimiser l'*expected cost function* $f(\mathbf{D})$ définie par :

$$f(\mathbf{D}) = \mathbb{E}_x[l(x, \mathbf{D})] = \lim_{n \rightarrow +\infty} f_n(\mathbf{D}) \text{ p-s} \quad (3.11)$$

où l'espérance (*expectation*) est calculée relativement à \mathbf{X} que nous pouvons considérer comme une variable aléatoire de loi de probabilité $p(x)$.

En particulier, considérant un *train dataset* \mathbf{X} de taille finie, chercher à obtenir un résultat précis sur $f_n(\mathbf{D})$ n'est pas intéressant puisqu'il s'agit seulement d'une approximation de $f(\mathbf{D})$. Il faudra alors considérer le *train dataset* \mathbf{X} comme une variable aléatoire \mathcal{X} .

Chapter 4

Online Dictionary Learning

Nous présentons dans ce chapitre les composantes principales d'un algorithme itératif *on-line* que nous avons choisi d'implémenter pour résoudre le problème de *dictionary learning*.

4.1 Algorithme

L'algorithme d'*online dictionary learning* est résumé dans **Algorithm 1**.

Tout d'abord, nous devons nous procurer un *dataset* \mathbf{X} composées de n *features* x_i exprimées par p *parameters* - que nous considérerons d'un point de vue probabiliste comme étant un échantillon de n observations i.i.d issues d'une variable aléatoire \mathcal{X} de loi $p(x)$.

Nous devons aussi nous procurer un *dictionary* initial \mathbf{D} qui sera ici notre référence d'apprentissage, de préférence issu de l'ensemble \mathcal{C} - le choix du dictionnaire de base pouvant avoir certainement un impact dans l'apprentissage.

Considérons alors la simulation d'une réalisation x_t de la variable \mathcal{X} à l'instant t - en choisissant uniformément une observation x_i parmi l'ensemble des observations de \mathbf{X} . Notre souhait est de proposer une mise à jour optimale du dictionnaire \mathbf{D} connaissant la réalisation x_t . Cette procédure est similaire à la méthode de *stochastique gradient descent*.

Ainsi, nous proposons de réaliser une boucle sur un ensemble de réalisations x_t (autant de réalisations que nous le souhaitons pour l'apprentissage) pour mettre à jour le dictionnaire \mathbf{D} , alternant deux étapes de résolution de problèmes d'optimisation.

La première étape consiste en la résolution du *sparse coding problem* par la méthode du **LARS-Lasso** pour trouver la décomposition r_t de x_t étant donné le dictionnaire \mathbf{D}_{t-1}

obtenu à l'itération précédente, par minimisation de la *cost function* :

$$f r_t(r) = \frac{1}{2} \|x_t - \mathbf{D}_{t-1} r\|_2^2 + \lambda \|r\|_1 \quad (4.1)$$

La deuxième étape consiste en la résolution du *dictionary update problem* par la méthode du **block-coordinate descent with warm restarts** pour calculer le nouveau dictionnaire \mathbf{D}_t dans l'ensemble \mathcal{C} connaissant toutes les précédentes décompositions r_i pour $i \in [1, t]$ obtenues lors des étapes précédentes de l'algorithme, par minimisation de la *cost function* :

$$f d_t(\mathbf{D}) = \frac{1}{t} \sum_{i=1}^t \frac{1}{2} \|x_i - \mathbf{D} r_i\|_2^2 + \lambda \|r_i\|_1 \quad (4.2)$$

L'intérêt de cette approche est de pouvoir approximer efficacement l'*expected cost function* $f(\mathbf{D})$ (Eq.3.9) par convergence de l'*empirical cost function* $f_t(\mathbf{D})$ (Eq.3.4) d'après des propriétés de convergence déjà démontrées par des experts - et ainsi obtenir un dictionnaire \mathbf{D}_T "bon" pour représenter \mathbf{X} .

Algorithm 1 Online Dictionary Learning

Require: $x \in \mathbb{R}^p \sim p(x)$ (variable aléatoire simulée à travers un algorithme générant des échantillons i.i.d de p),

$\lambda \in \mathbb{R}$ (paramètre de régularisation),

$D_0 \in \mathbb{R}^{p \times k}$ (dictionnaire initial),

T (nombre d'itération),

B (taille des batchs).

$A_0 \leftarrow 0, B_0 \leftarrow 0$

for $t = 1$ to T **do**

Extrait x_t de $p(x)$.

Sparse coding problem : Résolution **LARS-Lasso**

$$r_t = \underset{r \in \mathbb{R}^k}{\operatorname{argmin}} \frac{1}{2} \|x_t - \mathbf{D}_{t-1} r\|_2^2 + \lambda \|r\|_1$$

$$A_t \leftarrow A_{t-1} + r_t r_t^T$$

$$B_t \leftarrow B_{t-1} + x_t r_t^T$$

Dictionary update problem : Résolution **block-coordinate descent with warm restarts**

$$\mathbf{D}_t = \underset{\mathbf{D} \in \mathcal{C}}{\operatorname{argmin}} \frac{1}{t} \sum_{i=1}^t \frac{1}{2} \|x_i - \mathbf{D} r_i\|_2^2 + \lambda \|r_i\|_1$$

end for

return \mathbf{D}_t (learned dictionary)

4.2 Sparse coding problem

Le *sparse coding problem* 4.1 pour un dictionnaire \mathbf{D} donné consiste en la résolution d'un l_1 -regularized linear least-squares problem. La méthode **LARS-Lasso** permet justement de résoudre ce problème. La régression par moindre angle (*Least-angle regression* - **LARS**) est un algorithme permettant d'ajuster des modèles de régression linéaire à des données de grande dimension.

Supposons que nous nous attendions à ce qu'une variable de réponse - ici il s'agit de la réalisation x_t - soit déterminée par une combinaison linéaire d'un sous-ensemble de covariables potentielles - ici il s'agit de l'ensemble des *atoms* du dictionnaire \mathbf{D} . Alors, l'algorithme **LARS** fournit un moyen de produire une estimation des variables à inclure, ainsi que leurs coefficients - ici il s'agit de la décomposition r_t de x_t sachant \mathbf{D} .

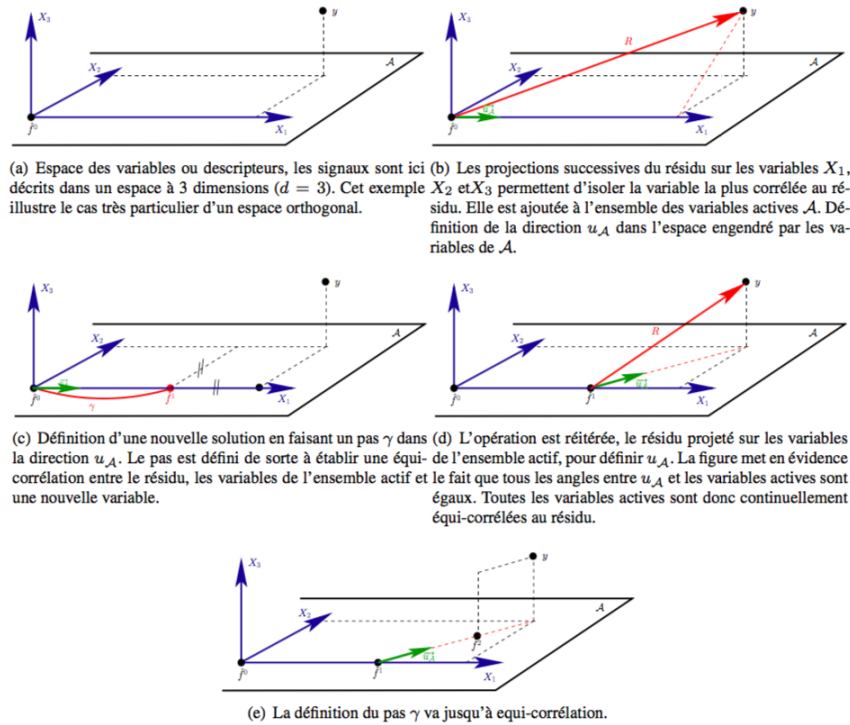


Figure 4.1: Explication graphique du fonctionnement de l'algorithme

Au lieu de donner un résultat vectoriel, la solution **LARS** consiste en une courbe désignant la solution pour chaque valeur de la norme l_1 - ou pénalisation Lasso - du vecteur de paramètres r_t (et donc de même pour chaque valeur du paramètre de régularisation λ). L'algorithme est similaire à la régression *forward stepwise*, mais au lieu d'inclure des variables à chaque étape, les paramètres estimés sont augmentés dans une direction équiangulaire à leurs corrélations avec le résidu (voir explication graphique).

4.3 Dictionary update problem

L'algorithme de mise à jour du dictionnaire \mathbf{D} utilise la méthode de *block-coordinate descent with warm restarts* et est présenté dans l'**Algorithm 2**.

La méthode générique *coordinate descent* est basée sur l'idée que la minimisation d'une fonction multivariée peut être obtenue en la minimisant dans une direction à la fois - c'est-à-dire résoudre des problèmes d'optimisation univariés - ou du moins beaucoup plus simples - à travers une boucle sur toutes les variables.

Dans le cas le plus simple de la *cyclic coordinate descent*, nous effectuons une itération cyclique à travers les directions x_1, \dots, x_d une à la fois, en minimisant la fonction objectif $F(x_1, \dots, x_d)$ par rapport à chaque direction de coordonnées.

Dans le cas d'une minimisation selon un dictionnaire $\mathbf{D} = (d_1, \dots, d_k)$, la méthode *block-coordinate descent* consiste à minimiser le problème 4.2 par blocs de coordonnées d_1, \dots, d_k .

Algorithm 2 Dictionary Update

Require: $\mathbf{D} = [d_1, \dots, d_k] \in \mathbb{R}^{p \times k}$ (dictionnaire),

$$\mathbf{A} = [a_1, \dots, a_k] \in \mathbb{R}^{k \times k} = \sum_{i=1}^t r_i r_i^T,$$

$$\mathbf{B} = [b_1, \dots, b_k] \in \mathbb{R}^{m \times k} = \sum_{i=1}^t x_i r_i^T$$

repeat

for $j = 1$ to k **do**

 Mettre à jour la j -ème colonne de \mathbf{D} d'après le *Dictionary update problem* :

$$\mathbf{D}_t = \underset{\mathbf{D} \in \mathcal{C}}{\operatorname{argmin}} \frac{1}{t} \sum_{i=1}^t \frac{1}{2} \|x_i - \mathbf{D} r_i\|_2^2 + \lambda \|\mathbf{D}\|_1 = \underset{\mathbf{D} \in \mathcal{C}}{\operatorname{argmin}} \frac{1}{t} \left(\frac{1}{2} \operatorname{Tr}(\mathbf{D}^T \mathbf{D} \mathbf{A}) - \operatorname{Tr}(\mathbf{D}^T \mathbf{B}) \right)$$

$$u_j = \frac{1}{\mathbf{A}_{j,j}} (b_j - \mathbf{D} a_j) + d_j$$

$$d_j = \frac{1}{\max(\|u_j\|_2, 1)} u_j$$

end for

until convergence

L'un des avantages de cette méthode d'apprentissage est qu'elle ne nécessite aucun paramétrage - c'est à dire qu'un *learning rate* ou taux d'apprentissage n'est pas utilisé pour l'apprentissage.

Concrètement, la méthode *block-coordinate descent with warm restarts* met à jour de manière séquentielle les colonnes du dictionnaire \mathbf{D} toutes choses égales par ailleurs.

Démonstration de la mise à jour

Nous cherchons le dictionnaire \mathbf{D}_t dans l'ensemble \mathcal{C} connaissant toutes les précédentes décompositions r_i pour $i \in [1, t]$ obtenues lors des étapes précédentes de l'**Algorithme 1**, par minimisation de la *cost function* :

$$\mathbf{D}_t = \underset{\mathbf{D} \in \mathcal{C}}{\operatorname{argmin}} f d_t(\mathbf{D}) = \underset{\mathbf{D} \in \mathcal{C}}{\operatorname{argmin}} \frac{1}{t} \sum_{i=1}^t \frac{1}{2} \|x_i - \mathbf{D} r_i\|_2^2 + \lambda \|r_i\|_1 \quad (4.3)$$

En développant l'expression de $f d_t$ et en ne gardant que les termes dépendants de \mathbf{D} , nous obtenons une expression plus simple à résoudre que nous pouvons simplifier d'après des propriétés matricielles :

$$\underset{\mathbf{D} \in \mathcal{C}}{\operatorname{argmin}} \sum_{i=1}^t \frac{1}{2} (\mathbf{D} r_i)^T \mathbf{D} r_i - x_i^T \mathbf{D} r_i \Rightarrow \underset{\mathbf{D} \in \mathcal{C}}{\operatorname{argmin}} F(\mathbf{D}) = \frac{1}{2} \operatorname{Tr}(\mathbf{D}^T \mathbf{D} \mathbf{A}) - \operatorname{Tr}(\mathbf{D}^T \mathbf{B}) \quad (4.4)$$

$$\underset{\mathbf{D} \in \mathcal{C}}{\operatorname{argmin}} F(\mathbf{D}) = \underset{\mathbf{D} \in \mathcal{C}}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^k d_i^T \mathbf{D} a_i - \sum_{i=1}^k d_i^T b_i \quad (4.5)$$

Nous cherchons désormais la direction de plus forte pente de $F(\mathbf{D})$ (et donc de $f d_t(\mathbf{D})$) pour une coordonnée d_j donnée qui peut être obtenue en calculant la dérivée partielle de $F(\mathbf{D})$ par rapport à d_j ainsi que le pas de la descente ¹ :

$$\frac{\partial F(\mathbf{D})}{\partial d_j} = \mathbf{D} a_j - b_j \quad | \quad \alpha_j = \frac{1}{A_{j,j}} \quad (4.6)$$

Ainsi, il suffit de mettre à jour successivement les *atoms* ou colonnes du dictionnaire \mathbf{D} pendant un nombre d'itération voulu pour atteindre la convergence :

$$\forall j \in [1, k], d_j^{(l+1)} = d_j^{(l)} - \alpha_j \frac{\partial F(\mathbf{D})}{\partial d_j} \quad | \quad d_j^{(l+1)} \leftarrow \frac{1}{\max(1, \|d_j^{(l+1)}\|_2)} d_j^{(l+1)} \quad (4.7)$$

Pour toujours s'assurer que le dictionnaire \mathbf{D} appartient à \mathcal{C} , il est nécessaire de normaliser les colonnes d_j si leur norme est trop grande.

¹Au cours de mes recherches, je n'ai pas réussi à savoir si le choix de α_j était justifié ou arbitraire. Empiriquement, les résultats sont concluants rendant légitime ce choix.

Discussion

Puisque ce problème d'optimisation convexe admet des contraintes séparables dans les blocs mis à jour (colonnes), la convergence vers un optimum global est garantie (d'après Bertsekas).

En pratique, puisque les vecteurs r_i sont parcimonieux, les coefficients de la matrice \mathbf{A} sont en général concentrés sur la diagonale, ce qui rend la descente des coordonnées par blocs plus efficace.

Puisque l'algorithme utilise la valeur de \mathbf{D}_{t-1} pour le calcul de \mathbf{D}_t , une seule itération s'est révélée empiriquement suffisante.

D'autres approches ont été proposées pour mettre à jour \mathbf{D} . Par exemple, il a été suggéré d'utiliser une méthode de Newton sur le dual de l'équation. Mais cela nécessite d'inverser une matrice $k \times k$ à chaque itération, ce qui n'est pas pratique pour un algorithme *online*.

Chapter 5

Validation expérimentale

Nous présentons dans ce chapitre l'expérimentation de notre algorithme *online dictionary learning* et les résultats obtenus sur la base d'apprentissage **MNIST**.

5.1 Base d'apprentissage MNIST



Figure 5.1: Visualisation d'un échantillon d'images issu de la base d'apprentissage MNIST

La base de données **MNIST** pour *Modified National Institute of Standards and Technology*, est une base de données de chiffres écrits à la main. C'est un jeu de données très utilisé en apprentissage automatique.

La reconnaissance de l'écriture manuscrite est un problème difficile, et un bon test pour les algorithmes d'apprentissage. La base **MNIST** est devenue un test standard. Elle regroupe 60000 images d'apprentissage et 10000 images de test. Ce sont des images en noir et blanc de 28 pixels de côté soit 784 pixels au total.

A l'image des applications des traitements du signal et d'images que nous avons pu présenter en introduction, dans une volonté de modéliser les chiffres manuscrits comme une combinaison parcimonieuse de quelques *images de base* contenant des structures sous-jacentes caractéristiques des chiffres manuscrits, nous considérons ici le problème de réduction de la dimensionnalité des images, dont l'objectif est de réduire le nombre de variables de base au nombre de 784 pixels à un nombre de variables significatives plus raisonnable.

La résolution par l'apprentissage non supervisé avec dictionnaire se prête parfaitement à ce problème. Les intérêts d'une telle démarche résident dans différents points :

- Le dictionnaire ainsi entraîné permettra de faire ressortir les structures internes de la base d'apprentissage **MNIST**.
- Cette méthode permettra aussi de décomposer un chiffre manuscrit en utilisant le dictionnaire. Cette décomposition peut être utile dans le traitement des images comme pour le débruitage, la classification ou la compression d'images.
- Cette méthode se prête aussi à l'apprentissage semi-supervisé où les caractéristiques apprises à travers le dictionnaire pourront être utilisées pour améliorer les performances dans un environnement supervisé.

5.2 Initialisation du dictionnaire

Pour réaliser nos expériences, nous avons proposé différentes méthodes d'initialisation du dictionnaire pour comparer leur efficacité dans l'apprentissage :

- D_1 (`make_dico_1`) : Cette méthode retourne un dictionnaire de taille $p \times k$ dont chacun de ses k *atoms* a été généré de manière aléatoire - c'est à dire que chaque feature d'un *atom*, qui ici représente un pixel de l'image, est choisi selon une gaussienne centrée réduite.
- D_2 (`make_dico_2`) : Cette méthode permet de construire un dictionnaire dont les *atoms* sont tirés aléatoirement de la base d'entraînement **MNIST**.
- D_3 (`make_dico_3`) : Cette méthode permet de construire un dictionnaire dont les *atoms* possèdent uniquement un carré de 9 pixels disposé aléatoirement.
- D_4 (`make_dico_4`) : Cette méthode permet de construire un dictionnaire composé de chaque chiffre manuscrit en nombre égal de la base d'entraînement **MNIST**.

5.3 Construction du modèle

Nous avons décidé d'implémenter notre modèle d'apprentissage sous le langage **R** en choisissant les paramètres suivants :

- **X** la base d'entraînement **MNIST** de taille $(n \times p) = (10000 \times 784)$
- λ le paramètre de régularisation égal à 2
- **D** le dictionnaire initiale issue de la méthode D_4 de taille $(p \times k) = (784 \times 100)$
- Tt le nombre d'itérations total pour la mise à jour égal à 1000¹
- Tb la taille des *batches* lors de la mise à jour égale à 1
- Tc le nombre d'itérations pour la convergence égal à 1

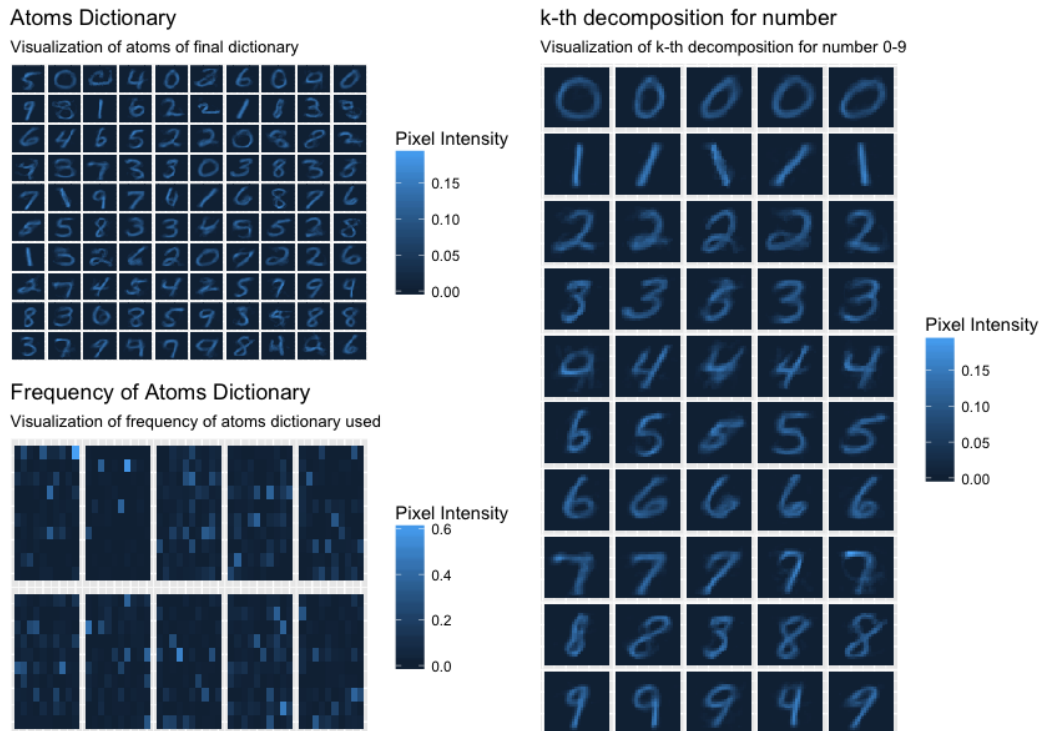


Figure 5.2: Visualisation du dictionnaire, des fréquences d'apparition des *atoms* dans la décomposition des chiffres manuscrits ainsi que leurs 5 premiers représentants les plus utilisés

¹Le choix du nombre d'itérations a été choisi de telle sorte à pouvoir obtenir des résultats en un temps raisonnable. En pratique, ce paramètre doit être relativement élevé pour pouvoir bien entraîner le dictionnaire.

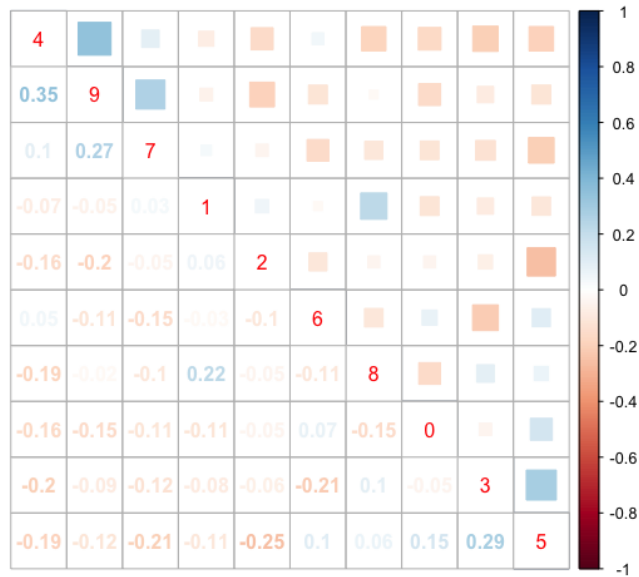


Figure 5.3: Visualisation de la corrélation des décompositions des chiffres manuscrits

5.4 Visualisation des résultats

Nous avons notamment proposé des outils de visualisation permettant de mieux appréhender :

- les 100 *atoms* du dictionnaire après apprentissage (fig 5.2 en haut à gauche).
- la fréquence d'apparition des *atoms* dans la décomposition des chiffres manuscrits (fig 5.2 en bas à gauche) - chaque matrice correspond à la matrice d'apparition des *atoms* du dictionnaire pour chaque chiffre manuscrit de 0 à 9, de gauche à droite et de haut en bas.
- leurs 5 premiers représentants les plus utilisés dans leurs décompositions (fig 5.2 à droite).
- la corrélation des décompositions des chiffres manuscrits (fig 5.3).

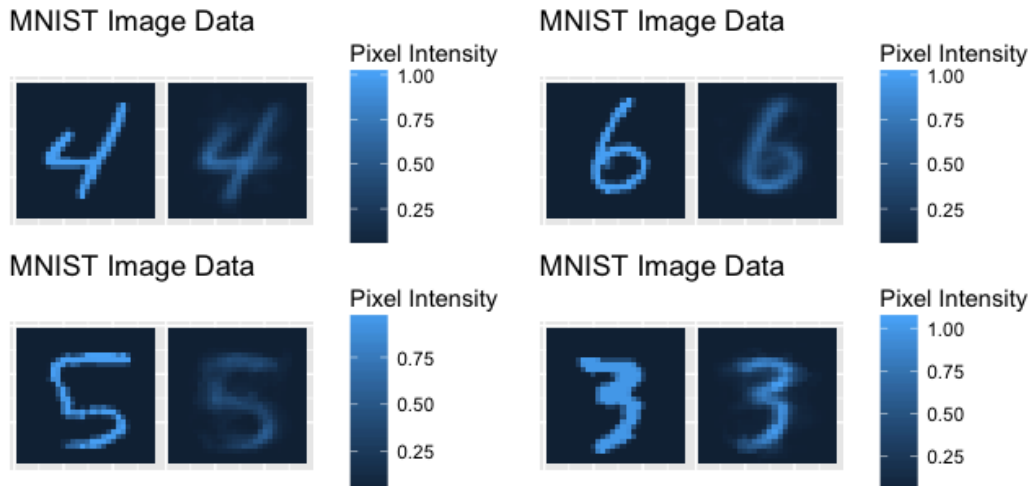


Figure 5.4: Visualisation de la reconstruction des chiffres manuscrits à partir de leur décomposition obtenue grâce au dictionnaire

5.5 Analyse des résultats

A partir de la visualisation de nos résultats numériques et graphiques, nous proposons de réaliser une analyse sur les performances de la méthode.

Nous remarquons alors que le dictionnaire entraîné a été capable de faire ressortir certaines structures internes de la base d'apprentissage **MNIST**.

Pour donner un exemple, à partir de la visualisation des principaux représentants dans la décomposition des chiffres, nous pouvons comprendre que le chiffre 1 peut être représenté principalement par des traits qui peuvent être soit verticaux, soit diagonaux.

Il faut par ailleurs souligner que l'émergence de ces structures n'est pas dépendante du choix de la méthode d'initialisation du dictionnaire D_4 . Ce choix influence essentiellement la vitesse de convergence vers ces structures, si celui-ci a été bien préparé, et donc également la performance de la méthode.

De plus, dans un environnement supervisé, nous pouvons considérer les corrélations entre les décompositions des chiffres manuscrits pour connaître à l'avance certaines structures significativement discriminantes.

Nous remarquons alors que les décompositions du chiffre 9 sont fortement corrélées avec celles du 4 et du 7 (0.35 et 0.27 respectivement) révélant que leurs écritures manuscrites sont relativement semblables. Inversement, les décompositions des chiffres 5 et 2 semblent décorréliées (0.25) indiquant que certaines structures peuvent permettre de les différencier.

Par ailleurs, il est possible d'identifier des structures qui sont uniquement utilisées par certains chiffres à partir des matrices de fréquences d'apparition des *atoms* dans leurs décompositions.

Enfin, la méthode nous permet de décomposer un chiffre manuscrit en utilisant le dictionnaire entraîné et ainsi le compresser (le réduire) à travers ses représentants principaux.

Nous proposons ici (fig 5.4) des reconstructions de certains chiffres à partir de leur décomposition.

Il est intéressant de remarquer que les décompositions des chiffres manuscrits sont parcimonieuses comme désiré - plus de 90% des coefficients des décompositions sont nuls.

5.6 Limites de la méthode

Le *dictionary learning* présente cependant certaines limites lors de son utilisation.

En effet, la méthode exige de fixer arbitrairement la valeur du paramètre de régularisation λ qui détermine l'importance relative entre l'erreur réalisée sur la représentation et la parcimonie de la décomposition. Ce choix est donc laissé à l'appréciation de l'utilisateur.

De plus, un apprentissage non-supervisé ne sera pas aussi efficace qu'un apprentissage supervisé dans un contexte de classification. Dans notre cas, certains chiffres comme le 4 et le 9 ne sont pas fortement distinguables ce qui nuit à la performance de la méthode. Cependant, des méthodes de *supervised dictionary learning* existent déjà et obtiennent de bonnes performances en classification.

Comme nous l'avons dit précédemment, l'initialisation du dictionnaire est crucial puisqu'il va permettre d'améliorer les performances de la méthode. Il est donc indispensable de bien choisir les *atoms* du dictionnaire pour obtenir une convergence rapide.

Chapter 6

Conclusion

Les techniques de *feature learning* sont efficaces dans la recherche de représentations nécessaires à la détection ou à la classification de caractéristiques, notamment dans les domaines du traitement du signal et de l'image.

Le *dictionary learning* est par ailleurs performant dans la recherche de représentation parcimonieuse dans un contexte d'apprentissage supervisé ou non supervisé selon les données à disposition et les objectifs fixés et est donc la solution idéale pour la résolution de problèmes de *sparse coding*.

Le caractère *online* de l'algorithme présenté dans ce rapport a l'avantage de proposer une méthode permettant d'utiliser un dictionnaire mis à jour en temps réel pouvant ainsi être utilisé en parallèle, ce qui peut être d'un grand intérêt dans certaines applications comme le traitement de vidéos.

Dans l'approfondissement de ce projet, il aurait été intéressant de mettre l'accent sur des méthodes de recherche d'hyper-paramètres optimaux pour améliorer la performance de l'algorithme d'*online dictionary learning*. En considérant, par exemple, différentes tailles de *batches* pour la mise à jour du dictionnaire, nous pouvons imaginer que la qualité de la convergence pourrait en être nettement améliorée.