Mean-shift : Clustering scalable et distribué

Gaël Beck, Hanane Azzag, Mustapha Lebbah, Tarn Duong, Christophe Cérin

Laboratoire d'Informatique de Paris Nord (LIPN)
Université Paris Nord – Paris 13, F-93430 Villetaneuse, France
Email: {beck, prénom.nom}@lipn.univ-paris13.fr

Résumé. Nous présentons dans ce papier un nouvel algorithme Mean-Shift utilisant les K-plus proches voisins pour la montée du gradient (NNMS : Nearest Neighbours Mean Shift). Le coût computationnel intensif de ce dernier a longtemps limité son utilisation sur des jeux de données complexes où un partitionnement en clusters non ellipsoïdaux serait bénéfique. Or, une implémentation scalable de l'algorithme ne compense pas l'augmentation du temps d'exécution en fonction de la taille du jeu de données en raison de sa complexité quadratique. Afin de pallier, ce problème nous avons introduit le "Locality Sensitive Hashing" (LSH) qui est une approximation de la recherche des K-plus proches voisins ainsi qu'une règle empirique pour le choix du K. La combinaison de ces améliorations au sein du NNMS offre l'opportunité d'un traitement pertinent aux problématiques du clustering appliquée aux données massives.

1 Introduction

L'objectif de la recherche non supervisée est d'affecter un label à des points non labélisés où le nombre et l'emplacement des clusters sont inconnus. Nous nous sommes concentrés sur un algorithme de clustering modal où le nombre de clusters est défini en terme de modes locaux de la fonction de densité de probabilité qui génère les données. Le plus connu des algorithmes de clustering modal est le k-means. Comme ce dernier est basé sur la distribution de mélange normale, il est contraint à trouver des clusters ellipsoidaux ce qui peut être inapproprié pour des jeux de données complexes. Le Mean-shift est une généralisation du k-means en raison de sa capacité à calculer des clusters de topologie aléatoire définis comme les bassins d'attractions des modes locaux générés par la montée de gradient de (Fukunaga et Hostetler, 1975). Afin de calculer les chemins de la montée de gradient, les k plus proches voisins sont appropriés car ils s'adaptent à la topologie locale des données. La version actuelle des k plus proches voisins Mean-shift contient des goulots d'étranglement posés par une grille de recherche multiple pour le choix d'un nombre de voisins optimal et par le calcul exact des k plus proches voisins. Nous proposons ici un nouvel algorithme qui résout ces gouffres computationnels : (a) une échelle normale efficace du choix du nombre des plus proches voisins qui évite la recherche en grille, (b) le locality sensitive hashing (LSH) qui est une version approximée des k plus proches voisins et (c) une implémentation MapReduce distribuée.

2 Méthode

2.1 Le Mean-shift

Le Mean Shift introduit par (Fukunaga et Hostetler, 1975), génère pour un point x de dimension d une séquence de points qui suivent le chemin en montée de la densité de gradient en utilisant la relation de récurrence :

$$\boldsymbol{x}_{j+1} = \frac{1}{k} \sum_{\boldsymbol{X}_i \in k\text{-nn}(\boldsymbol{x}_j)} \boldsymbol{X}_i \tag{1}$$

où $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ est un échantillon aléatoire obtenu d'une fonction de densité commune f, les k plus proches voisins de $\boldsymbol{x}$ sont k-nn$(\boldsymbol{x}) = \{\boldsymbol{X}_i : \|\boldsymbol{x} - \boldsymbol{X}_i\| \leq \delta_{(k)}(\boldsymbol{x})\}$ tel que $\delta_{(k)}(\boldsymbol{x})$ est la distance du k-ème plus proche voisin. $\boldsymbol{x}_0 = \boldsymbol{x}$. L'équation (1) donne au Mean Shift son nom en raison du déplacement successif des itérations de $\boldsymbol{x}_j$ vers la moyenne de ses k plus proches voisins pour la prochaine itération $\boldsymbol{x}_{j+1}$. La convergence de la séquence $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots\}$ vers un mode local pour la version à noyau de l'équation (1) a été établie par (Comaniciu et Meer, 2002) pour une large classe de noyaux sur des fenêtres fixées. Cette convergence reste valide quand la fenêtre fixe est remplacée par la distance des plus proches voisins qui décroit avec l'augmentation du nombre d'itérations.

Le chemin de montée de gradient vers les modes locaux produit par l'équation (1) forme les bases de l'Algorithme 2.1 (NNMS), notre méthode des plus proches voisins Mean-shift. Les entrées du NNMS sont les échantillons de données $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ et les points candidats que nous souhaitons clusteriser $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ (Ils peuvent être $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ mais ce n'est pas un prérequis). Les paramètres de réglage sont les suivants :

- le nombre de plus proches voisins k
- le seuil sous lequel la convergence des itérations est considérée comme étant sufisante ε_1
- le nombre maximum d'itérations $j_{\max}$
- le seuil sous lequel deux itérés finaux sont considérés comme étant membres du même cluster ε_2
- la cardinalité minimale des clusters formés $c_{\min}$

Les sorties sont les labels des clusters des points candidats $\{\mathrm{c}(\boldsymbol{x}_1), \ldots, \mathrm{c}(\boldsymbol{x}_m)\}$. Il y a trois sous-routines à l'Algorithme 2.1. Les lignes 1-6 correspondent à la formation des chemins de la montée de gradient dans l'équation 1 qui sont itérés jusqu'à ce que la distance de la dernière itération soit infèrieure à ε_1 ou que le nombre maximum d'itérations $j_{\max}$ soit atteint. Les sorties de ces lignes sont les itérés finaux $\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_m^*$. Les lignes 7-8 concernent la fusion des itérés finaux dans le même cluster lorsque la distance les séparant est sous le seuil ε_2, ceci créant un regroupement initial des $\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_m^*$. Les lignes 9-13 déterminent si les plus petits clusters ont une cardinalité supérieure à $s_{\min}$ sinon on fusionne les clusters concernés avec leur voisin le plus proche pour produire $\mathrm{c}(\boldsymbol{x}_1^*), \ldots, \mathrm{c}(\boldsymbol{x}_m^*)$. La ligne 14 assigne les labels de ces clusters aux données originales $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$.

2.2 Choix du nombre de plus proches voisins suivant une échelle normale

Le paramètre de réglage critique pour le Mean shift est le choix du nombre de plus proches voisins k. Le travaux pionniers de (Loftsgaarden et Quesenberry, 1965), (Fukunaga et Hostetler,

Algorithm 1 NNMS – Plus proches voisins Mean-shift avec les exacts k plus proches voisins

Entrées : $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n\}, \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}, k, \varepsilon_1, \varepsilon_2, j_{\max}, s_{\min}$
Sorties : $\{\mathrm{c}(\boldsymbol{x}_1), \ldots, \mathrm{c}(\boldsymbol{x}_m)\}$
/* Calcul du chemin de montéé de gradient */
for $\ell := 1$ to m **do**
  $j := 0$; $\boldsymbol{x}_{\ell,0} := \boldsymbol{x}_\ell$;
  $\boldsymbol{x}_{\ell,1} :=$ mean of k-nn of $\boldsymbol{x}_{\ell,0}$;
  while $\|\boldsymbol{x}_{\ell,j+1}, \boldsymbol{x}_{\ell,j}\| > \varepsilon_1$ **or** $j < j_{\max}$ **do**
    $j := j + 1$; $\boldsymbol{x}_{\ell,j+1} :=$ mean of k-nn of $\boldsymbol{x}_{\ell,j}$;
  $\boldsymbol{x}^*_\ell := \boldsymbol{x}_{\ell,j}$;
/* Création des clusters par fusions des itérés finaux */
for $\ell_1, \ell_2 := 1$ to m **do**
  if $\|\boldsymbol{x}^*_{\ell_1} - \boldsymbol{x}^*_{\ell_2}\| \leq \varepsilon_2$ **then** $\mathrm{c}(\boldsymbol{x}^*_{\ell_1}) := \mathrm{c}(\boldsymbol{x}^*_{\ell_2})$;
/* Fusion des petits clusters */
$C^* :=$ cluster avec une cardinalité minimale ;
while $\mathrm{card}(C^*) < s_{\min}$ **do**
  $C' :=$ plus proche cluster à C^* ;
  for $\boldsymbol{x}^*_\ell \in C^*$ **do** $\mathrm{c}(\boldsymbol{x}^*_\ell) := \mathrm{c}(C')$;
  $C^* :=$ cluster avec une cardinalité minimale ;
for $\ell := 1$ to m **do** $\mathrm{c}(\boldsymbol{x}_\ell) := \mathrm{c}(\boldsymbol{x}^*_\ell)$;

1973) établissent l'erreur quadratique optimale des sélecteurs locaux et globaux pour les estimateurs de densité des plus proches voisins, sachant que ces auteurs ne considèrent pas les sélecteurs basés sur les données. Une grille de recherche basée sur les données cherche à minimiser les indices de qualité de recherche non supervisée comme l'indice Silhouette considéré par (Wang et al., 2007). Notre proposition d'échelle normale pour le sélecteur est :

$$k_{\mathrm{NS}} = v_0[4/(d+4)]^{d/(d+6)} n^{6/(d+6)} \tag{2}$$

où $v_0 = \pi^{d/2}\Gamma((d+2)/d))$ est l'hyper-volume d'une sphère unitaire d-dimentionnel. La dérivation de l'équation (2) est donnée dans (Duong et al., 2016). Elle suit l'assertion que la sélection des paramètres de réglages basés sur le gradient de densité plutôt que sur la densité elle-même est plus adéquate pour le mean shift (Chacón et Duong, 2013). La complexité de k_{NS} est $O(1)$ ce qui contraste avec le $O(n)$ de la grille de recherche pour sélectionner le nombre optimal de plus proches voisins k sachant que le nombre de recherches de valeurs possibles est usuellement réglé pour être proportionnel à n.

2.3 Plus proches voisins approximés avec le Locality Sensitive Hashing

La tâche calculatoire la plus intensive dans NNMS est le calcul des k plus proches voisins plutôt que la sélection du nombre de plus proches voisins. En effet, pour chaque point candidat, cela requiert le calcul et le tri de la distance $\|\boldsymbol{X}_i - \boldsymbol{x}_j\|, i = 1, \ldots, n, j = 1, \ldots, m$, qui est $O(mn \log n)$. Dans les cas usuels où m est du même ordre de grandeur que n, cela empêche son application pour des jeux de données importants. Une approche de réduction de complexité

prometteuse tient sur le calcul des approximés plus proches voisins plutôt que sur les exacts plus proches voisins. Parmi celles existantes, le locality sensitive hashing introduit par (Datar et al., 2004),(Datar et al., 2004) est une approche probabiliste basée sur une projection scalaire aléatoire de points multivariés $\boldsymbol{x}$

$$L(\boldsymbol{x}; w) = (\boldsymbol{Z}^T \boldsymbol{x} + U)/w$$

où $\boldsymbol{Z} \sim N(0, \mathbf{I}_d)$ est une variable aléatoire normale d-variée et $U \sim \mathrm{Unif}(0, w)$ est une variable aléatoire uniforme prise sur $[0, w)$, $w > 0$. Une table de hashage dont les blocs sont basés sur des valeurs entières $\lfloor L(\boldsymbol{X}_i; w) \rfloor, i = 1, \ldots, n$ est alors construite. En raison de propriétés statistiques de la distribution normale, les points proches dans l'espace multidimensionnel de départ auront tendance à tomber dans les mêmes blocs scalaires et les points distants tomberont dans des blocs différents comme vérifié dans (Slaney et Casey, 2008). D'importantes valeurs de w impliqueront moins de blocs avec plus de précision dans la préservation des caractéristiques de $\boldsymbol{X}_i$, tandis que de petites valeurs de w entraineront plus de blocs avec moins de précision. Nous avons préféré paramétriser le LSH par le nombre de blocs M de la table de hashage. Nous avons fixé $w = 1$ sans perte de généralité $L_i \equiv L(\boldsymbol{X}_i; 1)$. Ces projections scalaires sont ensuite triées dans leur ordre statistique $w = (L_{(n)} - L_{(1)})/M$ où $I_j = [L_{(1)} + w(j-1), L_{(1)} + wj]$, $j = 1, \ldots, M$. La valeur hashée de $\boldsymbol{x}$ est l'index de l'intervalle dans lequel $L(\boldsymbol{x}; 1)$ tombe

$$H(\boldsymbol{x}) = j\mathbf{1}\{L(\boldsymbol{x}; 1) \in I_j\} \tag{3}$$

où $\mathbf{1}\{\cdot\}$ est la fonction d'indication. Afin de chercher les approximés plus proches voisins, le réservoir des potentiels plus proches voisins est réglé à la valeur du bloc contenant la valeur de hashage. Ce réservoir est élargi si nécessaire par concaténation avec les blocs voisins. Les approximés k plus proches voisins de $\boldsymbol{x}$ sont les k plus proches voisins contenus dans le réservoir réduit $R(\boldsymbol{x})$: k-$\widetilde{\mathrm{nn}}(\boldsymbol{x}) = \{\boldsymbol{X}_i \in R(\boldsymbol{x}) : \|\boldsymbol{x} - \boldsymbol{X}_i\| \leq \delta_{(k)}(\boldsymbol{x})\}$ où $\delta_{(k)}(\boldsymbol{x})$ est la distance seuil des plus proches voisins à $\boldsymbol{x}$. L'erreur d'approximation dans les plus proches voisins à $\boldsymbol{x}$ induite par la recherche dans $R(\boldsymbol{x})$ plutôt que dans toutes les données est probabilistiquement contrôllée, voir (Slaney et Casey, 2008).

L'Algorithme 2 NNLSH est une approximation de la recherche des plus proches voisins avec le LSH et la fonction de hashage fourni par l'équation (3). Les entrées sont les échantillons de données $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$. Dans les lignes 2-6, pour chaque point candidat $\boldsymbol{x}_\ell$, les approximés k-plus proches voisins k-$\widetilde{\mathrm{nn}}(\boldsymbol{x}_\ell)$ sont calculés à partir du réservoir $R(\boldsymbol{x}_\ell)$.

La proposition où le NNLSH est intégrée au NNMS a été faite par (Cui et al., 2011), ce qui réduit la complexité à $O((mn/M)\log(n/M))$. Le nombre de blocs M est un paramètre de réglage crucial. Malgré un fort intérêt pour le LSH (Har-Peled et al., 2012), il n'existe pas de méthode optimale pour sélectionner le nombre de blocs, nous examinerons donc des heuristiques de performance dans la prochaine section.

Implémenter les approximatifs plus proches voisins NNMS de manière distribuée avec un processus maître et N processus esclaves réduit la complexité à $O(mn/(MN)\log(n/(MN)))$. C'est notre proposition, le DNNMS dans l'Algorithme 3. Les entrées et sorties sont les mêmes que pour l'Algorithme 1. Pour la j-ème itération, les chemins de montée de gradient $\boldsymbol{x}_j = [\boldsymbol{x}_{1,j}; \ldots; \boldsymbol{x}_{m,j}]$ sont collectés dans une matrice $m \times d$. Aux lignes 1 à 6, on itère jusqu'à une convergence globale $\|\boldsymbol{x}_{j+1} - \boldsymbol{x}_j\| \leq \epsilon_2 \equiv \|\boldsymbol{x}_{1,j+1} - \boldsymbol{x}_{1,j}\|, \ldots, \|\boldsymbol{x}_{m,j+1} - \boldsymbol{x}_{m,j}\| \leq \epsilon_2$ ou jusqu'au nombre maximal d'itérations $j_{\max}$. Certains calculs redondants sont effectués lorsque certains des $\boldsymbol{x}_{\ell,j}$ ont déjà convergé, mais cette forme de calcul est nécessaire pour une

parallélisation effective en MapReduce (Dean et Ghemawat, 2008). Le paradigme MapReduce est plus efficace si les algorithmes en séries sont repensés, passant d'une itération sur chaque candidat à une itération sur l'ensemble des candidats simultanément. Les lignes 7-14 décrivent la fusion des regroupements reprise de l'Algorithme 1 sans modification majeure étant donné que le MapReduce n'est pas requis ici.

Algorithm 2 NNLSH – Approximés k plus proches voisins avec LSH

Entrées : $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n\}, \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}, k, M$
Sorties : $\{k\text{-}\widetilde{\mathrm{nn}}(\boldsymbol{x}_1), \ldots, k\text{-}\widetilde{\mathrm{nn}}(\boldsymbol{x}_m)\}$
/* Création des tables de hashage avec M blocs */
1: **for** $i := 1$ to n **do** $H_i := H(\boldsymbol{X}_i)$;
/* Recherche des approximés plus proches voisins dans les blocs adjacents */
2: **for** $\ell := 1$ to m **do**
3: $R(\boldsymbol{x}_\ell) := \{\boldsymbol{X}_i : H_i = H(\boldsymbol{x}_\ell), i \in \{1, \ldots, n\}\}$
4: **while** $\mathrm{card}(R(\boldsymbol{x}_\ell)) < k$ **do**
5: $R(\boldsymbol{x}_\ell) := R(\boldsymbol{x}_\ell) \cup$ bloc adjacent ;
6: $k\text{-}\widetilde{\mathrm{nn}}(\boldsymbol{x}_\ell) := k$-nn de $R(\boldsymbol{x}_\ell)$ à $\boldsymbol{x}_\ell$;

Algorithm 3 DNNMS – Plus proches voisins Mean-shift distribué, avec approximés k plus proches voisins en utilisant le LSH

Entrées : $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n\}, \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}, k, \varepsilon_1, j_{\max}, \varepsilon_2, s_{\min}, M$
Sorties : $\{\mathrm{c}(\boldsymbol{x}_1), \ldots, \mathrm{c}(\boldsymbol{x}_m)\}$
/* Calcul des chemins de montée de gradient */
1: $j := 0$; $\boldsymbol{x}_0 := [\boldsymbol{x}_{1,0}; \ldots; \boldsymbol{x}_{m,0}]$;
2: $\boldsymbol{x}_1 :=$ mean of $k\text{-}\widetilde{\mathrm{nn}}$ of $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n\}$ to $\boldsymbol{x}_0$
3: **while** $\|\boldsymbol{x}_{j+1} - \boldsymbol{x}_j\| > \varepsilon_1$ **or** $j < j_{\max}$ **do**
4: $\boldsymbol{x}_{j+1} :=$ mean of $k\text{-}\widetilde{\mathrm{nn}}$ of $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n\}$ to $\boldsymbol{x}_j$;
/* cf Algorithme 2 */
5: $\boldsymbol{x}^* := [\boldsymbol{x}_{1,j}; \ldots; \boldsymbol{x}_{m,j}]$;
6: Identique aux lignes 7–14 dans l'Algorithme 2.1 ;

3 Résultats expérimentaux

3.1 Influence des paramètres de parallélisation

3.1.1 Nombre de noeuds

On peut observer sur la figure 1a que notre Scala/Spark implémentation de la montée de gradient avec les k-plus proches voisins est scalable, le temps d'exécution diminue efficacement avec les nombre de noeuds esclaves. Après investigations des lacunes de notre première implémentation, nous avons observer que l'étape de labélisation n'était pas scalable comme montré sur la figure 1b. Cependant, en réutilisant les LSH afin de segmenter les tâches comme

Algorithm 4 Labélisation distribuée avec les k plus proches voisins approximés

Entrées : $\{\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_m^*\}, M_2, \epsilon_2, \epsilon_3$
Sorties : $\{\mathrm{c}(\boldsymbol{x}_1), \ldots, \mathrm{c}(\boldsymbol{x}_m)\}$
/* Création des tables de hashages à M_2 blocs */
1: **for** $i := 1$ to m **do** $H_i := H(\boldsymbol{x}_i^*)$;
/* Labelisation des données */
2: **for** $\ell := 1$ to m **do**
3: $\quad R(\boldsymbol{x}_\ell^*) := \{\boldsymbol{x}_i^* : H_i = H(\boldsymbol{x}_\ell^*), i \in \{1, \ldots, m\}\}$
4: $\quad$ **for** $\ell_1, \ell_2 := 1$ to $card(R(\boldsymbol{x}_\ell^*))$ **do**
5: $\quad\quad$ **if** $\|\boldsymbol{x}_{\ell_1}^* - \boldsymbol{x}_{\ell_2}^*\| \leq \varepsilon_2$ **then** $\mathrm{c}(\boldsymbol{x}_{\ell_1}^*) := \mathrm{c}(\boldsymbol{x}_{\ell_2}^*)$;
/* Calcul des barycentres */
$C_1, ..., C_j$:= barycentre des clusters
/* Fusion des plus proches cluster */
6: **for** $C_1, C_2 := 1$ to j **do**
7: $\quad$ **if** $\|C_1 - C_2\| \leq \epsilon_3$ **then** Fusion de C_1 et C_2

décrit dans l'algorithme 4, nous avons observé une scalabilité de la solution réprésentée sur la figure 1c. La troisième étape consistant à fusionner les petits clusters avec leurs plus proches voisins se déroule localement sur le noeud maître, elle prend les coordonnées des barycentres et la cardinalité du cluster associé pour sortir une labélisation finale qui sera appliquée en parallèle. En pratique si les valeurs ϵ_2 et ϵ_3 sont bien choisies, cette étape est immédiate, un mauvais choix de ces valeurs peut entrainer la génération d'un grand nombre de clusters et faire exploser le temps d'exécution.

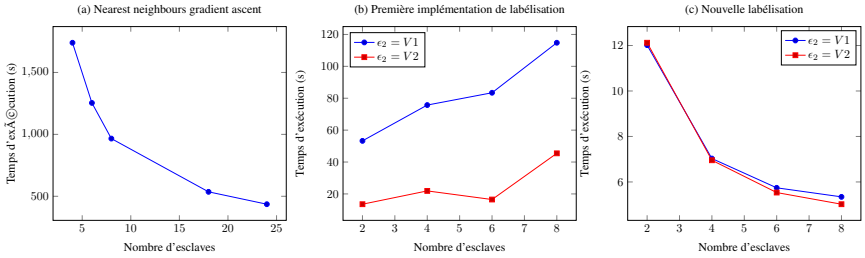

FIG. 1 – *Amélioration de scalabilité. (a) Montée de gradient KNN. (b) Première labélisation. (c) Nouvelle implémentation.*

3.1.2 Influence du nombre de blocs du LSH

Notre nouvelle implémentation tire avantage du LSH durant la phase de montée de gradient mais aussi pendant l'étape de labélisation. Un paramètre clé sera celui du nombre de blocs M_1, M_2 dans le LSH. Si on laisse ce dernier constant comme présenté sur la Figure 2, on constate que la complexité quadratique de la montée de gradient via les k plus proches voisins persiste tout comme pour l'étape de labélisation. Cependant, on peut observer sur la Figure 3

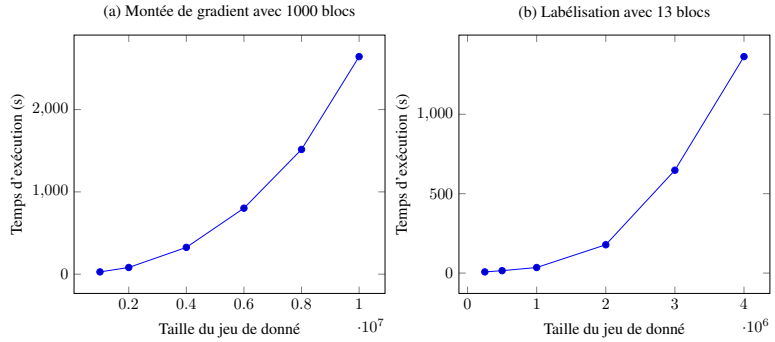

FIG. 2 – *Influence de la taille du jeu de données à nombre de bloc constant. (a) Sur la montée de gradient. (b) Sur la labélisation.*

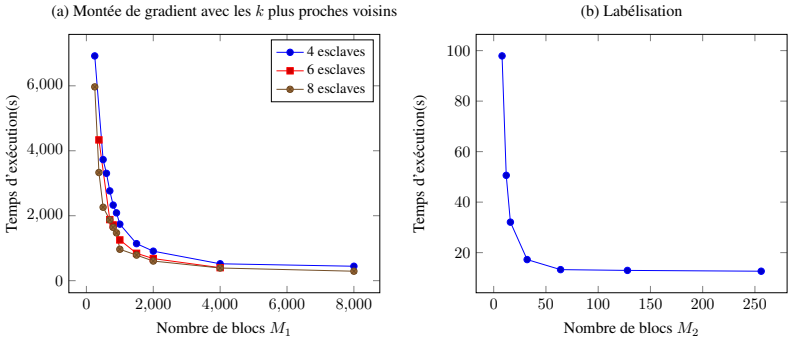

FIG. 3 – *Influence du nombre de blocs pour un jeu de données de taille fixe. (a) Sur la montée de gradient. (b) Sur la labélisation.*

le temps d'exécution diminuer rapidement, en fonction du nombre de blocs, puis ralentir pour atteindre un seuil qui dépendra du nombre de données d'entrées. Une observation intéressante concerne l'augmentation linéaire du temps d'exécution lorsqu'on fixe un nombre d'élément par bloc constant avec l'augmentation de la taille du jeu de données comme illustré sur la Figure 4. Ce résultat permet de conforter notre version du Mean Shift en tant qu'algorithme pleinement scalable.

3.2 Application à la segmentation d'image

La résurgence d'intérêt dans l'algorithme Mean-shift est dûe à son application à la segmentation d'image (Comaniciu, 2003) où une image est transformée dans un espace colorimétrique dans lequel chaque cluster correspond à des régions segmentées de l'image originale. L'espace 3-dimentionnel $L^*u^*v^*$ de couleur (Pratt, 2001) est un choix commun. Sachant qu'une image est un tableau bidimentionnel de pixels, disons que (x, y) sont les indices de lignes et de colonnes d'un pixel. Les informations spatiales et colorimétriques d'un pixel peuvent donc être concaténées en un vecteur 5-dimensionnel (x, y, L^*, u^*, v^*) dans la jointure des domaines

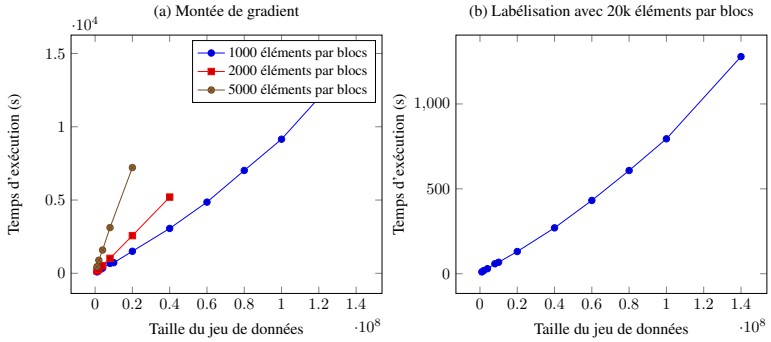

FIG. 4 – *Influence de la taille du jeu de données . (a) Sur la montée de gradient. (b) Sur la labélisation.*

spatio-colorimétriques. Afin d'illustrer cela, nous prenons l'image #171 du jeu d'entrainement coloré de Berkeley Segmentation Dataset and Benchmark [1]. Dans la Fig.3(a-b) on retrouve les pixels originaux RGB 481 × 321 de l'image JPEG et le scatter plot des $n = 154401$ jointure des coordonnées spatio-colorimétriques (x, y, L*, u*, v*).

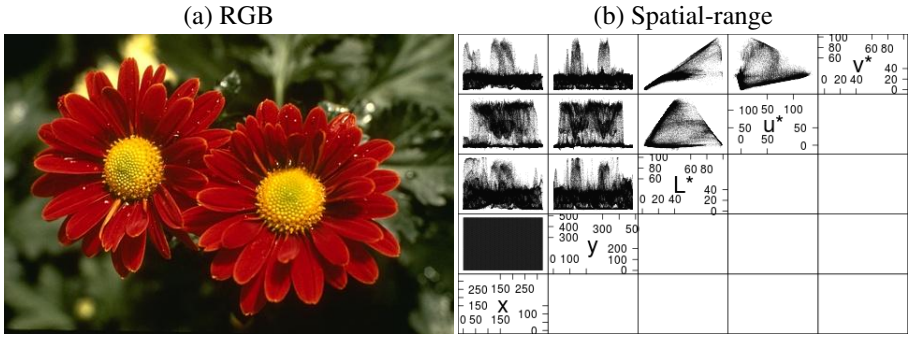

FIG. 5 – *Representations des couleurs de l'image. (a) Image RGB de 481×321 pixels. (b) Scatter plot avec* $n = 154401$ *transformé en espace* (x, y, L^*, u^*, v^*).

Un algorithme de segmentation d'image basé sur le Mean-shift à noyau avait été introduit dans [2] que nous avons adapté pour un usage avec NNMS. Les paramètres de réglages pour NNMS et DNNMS sont $k_{\mathrm{NS}} = 2463, \varepsilon_1 = 0.005$ fois la marge maximale de la portée des données, $j_{\max} = 100, \varepsilon_2 = 10\varepsilon_1, s_{\min} = 1544$. Nous exécutons le DNNMS-M avec $M = 200, 500, 1000$ blocs. Les temps d'exécution sont respectivement $20, 13$ et 10 minutes, une amélioration significative en comparaison à la nuit de calcul nécessaire pour le NNMS sur un ordinateur de bureau standard. Quant à la qualité de la segmentation d'images dans la Fig.4(a) pour le NNMS avec les exacts plus proches voisins. Cette image segmentée offre une considérable réduction de la complexité de l'image, tout en détectant les centres des fleurs incluant certains détails de granularité fine. Les contours, les bords des pétales, certaines ombres

1. http ://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds

ainsi que différents feuillages d'arrière-plan en sont la preuve. En raison de la nature aléatoire de la projection du LSH pour approximer les plus proches voisins dans le DNNMS, ces clusters sont moins compacts et plus diffus que ceux du NNMS où les projections LSH ne sont pas utilisées. Pour le DNNMS-200 dans la Fig.4(b) avec les approximés plus proches voisins avec $M = 200$ blocs, certains détails sont plus ou moins visibles en l'absence du LSH. Pour le DNNMS-500 et le DNNMS-1000 dans les Fig.4(c-d), les centres jaunes des fleurs sont moins clairement délimités pour les pétales, et il y a de considérables épanchements des pétales sur le feuillage. Nous observons que $M = 200$ blocs est un choix fortuit comme c'est aussi le cas dans la Fig.2 et de plus amples investigations sont requises pour un choix optimal général.

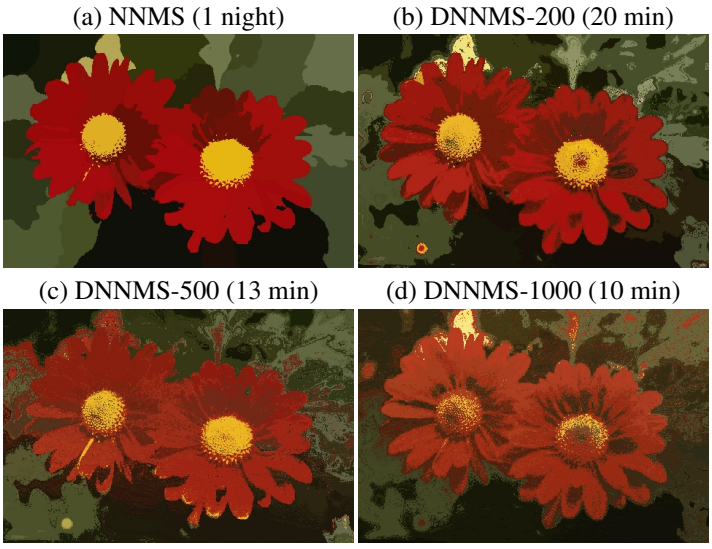

FIG. 6 – *Segmentation d'image colorée via les plus proches voisins Mean-shift. (a) NNMS avec le plus proche voisin exact Mean-shift en série. (b–d) DNNMS-M avec les approximatifs plus proches voisins Mean-shift distribué avec $M = 200, 500$ et 1000 blocs. Les temps d'exécutions sont respectivement 1 nuit, 20, 13 et 10 minutes.*

Le Berkeley Segmentation Dataset and Benchmark fournit une segmentation d'image humaine de leurs images à des fins de comparaisons. Dans la Fig.5(a-b) se trouvent deux détections de bordures faites par l'utilisateur #1107 et #1123. L'utilisateur #1107 se concentre sur la segmentation du feuillage d'arrière plan et le contour de la forme des fleurs, tout en ignorant les détails des pétales des fleurs. L'utilisateur #1123 quant Ã lui se concentre sur la segmentation des pétales individuelles dans le premier plan. Nous portons ici notre attention sur le NNMS et le DNNMS-200 (Fig. 5(c-d)). Le DNNMS-500 et le DNNMS-1000 (Fig. 5(e-f)) donne une qualité insuffisante de la détection des bords. Le NNMS et le DNNMS-200 sont capables de segmenter en une seule exécution avec un unique jeu de paramètres de réglage, simultanément le feuillage d'arrière plan et la forme des pétales de premier plan. On a ainsi une segmentation automatique combinant le résultat de deux experts humains se focalisant sur différentes zones de l'image.

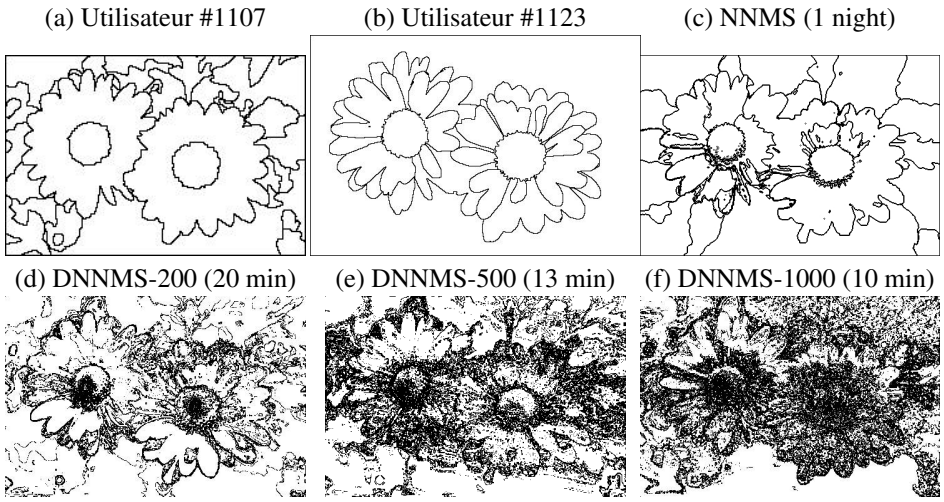

FIG. 7 – *Détection de bordure d'image segmentée. (a–b) Deux experts humains : utilisateur #1107 et #1123. (c) NNMS en série avec les exacts plus proches voisins. (d–f) DNNMS-M distribué avec les approximatifs plus proches voisins LSH avec $M = 200, 500$ et 1000 blocs.*

4 Conclusion

Nous avons introduit plusieurs améliorations à l'algorithme des plus proches voisins Mean-shift. La première est une heuristique du choix d'une valeur optimale du nombre de plus proches voisins. La seconde est l'emploi d'une approximation des plus proches voisins via le locality sensitive hashing pour la phase de montée de gradient mais aussi pour la phase de labélisation. La troisième est une implémentation dans un écosystème distribué. Nous avons démontré que ces améliorations diminuent drastiquement le temps d'exécution tout en maintenant la qualité du regroupement vis à vis des exacts plus proches voisins. Ces améliorations rendent possible l'application du Mean-shift pour le clustering appliqué au Big Data dans un futur proche. Certaines améliorations restent cependant à faire sur les paramètres de réglages cruciaux, i.e le nombre de plus proches voisins ainsi que le nombre de blocs dans le locality sensitive hashing pour les approximés plus proches voisins est requis.

Références

Chacón, J. E. et T. Duong (2013). Data-driven density estimation, with applications to nonparametric clustering and bump hunting. *Electronic Journal of Statistics 7*, 499–532.

Comaniciu, D. (2003). An algorithm for data-driven bandwidth selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 281–288.

Comaniciu, D. et P. Meer (2002). Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis And Machine Intelligence 24*, 603–619.

Cui, Y., K. Cao, G. Zheng, et F. Zhang (2011). An adaptive mean shift algorithm based on lsh. *Procedia Engineering 23*, 265–269.

Datar, M., N. Immorlica, P. Indyk, , et V. S. Mirrokni (2004). Locality-sensitive hashing scheme based on p-stable distributions. *Proceedings of the twentieth annual symposium on Computational geometry*, 253–262.

Dean, J. et S. Ghemawat (2008). MapReduce: Simplified data processing on large clusters. *Communications of the ACM 51*, 107–113.

Duong, T., G. B. H. Azzag, et M. Lebbah (2016). Nearest neighbour estimators of density derivatives, with application to mean shift clustering. *Pattern Recognition Letters 80*, 224–230.

Fukunaga, K. et L. Hostetler (1973). Optimization of k-nearest-neighbor density estimates. *IEEE Transactions on Information Theory 19*, 320–326.

Fukunaga, K. et L. Hostetler (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory 21*, 32–40.

Har-Peled, S., P. Indyk, et R. Motwani (2012). Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing 8*, 321–350.

Loftsgaarden, D. O. et C. P. Quesenberry (1965). A nonparametric estimate of a multivariate density function. *Annal of Mathematical Statistics 36*, 1049–1051.

Pratt, W. K. (2001). Digital Image Processing: PIKS Inside.

Slaney, M. et M. Casey (2008). Locality-sensitive hashing for finding nearest neighbors. *IEEE Signal Processing Magazine*, 128–131.

Wang, X., W. Qiu, et R. H. Zamar (2007). A non-parametric clustering method based on local shrinking. *Computational Statistics & Data Analysis 52*, 286–298.

Summary

We introduce an efficient distributed implementation of nearest neighbour mean shift clustering (NNMS). The computationally intensive nature of NNMS has so far restricted its application to complex data sets where a flexible clustering with non-ellipsoidal clusters would be beneficial. A parallel implementation of the standard serial NNMS algorithm on its own brings insufficient performance gains so we introduce two further algorithmic improvements: a normal scale (NS) choice of the optimal number of nearest neighbours, and locality sensitive hashing (LSH) to approximate nearest neighbour searches. Combining these improvements into a single distributed algorithm DNNMS offers the potential for an efficient method for Big Data Clustering.