Note du cours 6 de ML : Apprentissage non supervisé

Nicolas Baskiotis
MLIA, LIP6, Sorbonne Université

6 mai 2020

Résumé

Ces notes ne sont malheureusement pas un cours et ne peuvent remplacer complétement le cours magistral. Elles ont été écrites très rapidement, elles sont destinées à vous guider autant que possible dans la lecture des slides en soulignant le message important de chacun. J'ai essayé de donner des références pour tous les points techniques, n'hesitez pas à vous y reporter pour affiner votre compréhension.

1 Introduction

Dans les cours précédents, nous avons vu principalement des techniques d'apprentissage supervisé, c'est-à-dire quand on dispose d'un ensemble d'exemples d'apprentissage avec leur label. Lorsque les labels ne sont pas disponibles (ou que l'on ne s'y intéresse pas), on parle d'apprentissage non supervisé. La problématique principale de l'apprentissage non supervisé est le clustering (mais pas seulement !) : former une partitionement des exemples où chaque ensemble d'exemples (un cluster) est le plus "cohérent" au sens d'une similarité entre exemples. Des exemples de contexte où il est nécessaire de faire de l'apprentissage non supervisé : lorsque les labels ne sont pas disponibles (pas d'expertise humaine, ou trop cher d'annoter les exemples), lorsqu'il est impossible d'étiquetter raisonablement (photos ou image - on ne sait pas toujours ce que l'on cherche, profils d'utilisateurs, ...), trop de catégories non pertinentes (description de produits/mots-clés dans des boutiques en ligne), recherhe de sous-structure de données (analyse musicale, analyse de séries temporelles, ...). Le principe générique de l'apprentissage non supervisé est de rassembler tout ce qui se ressemble et de ne pas mettre ensemble tout ce qui est trop différent. C'est un objectif vague, fortement dépendant du domaine d'application, qui peut aboutir facilement à des résultats sans aucun sens !

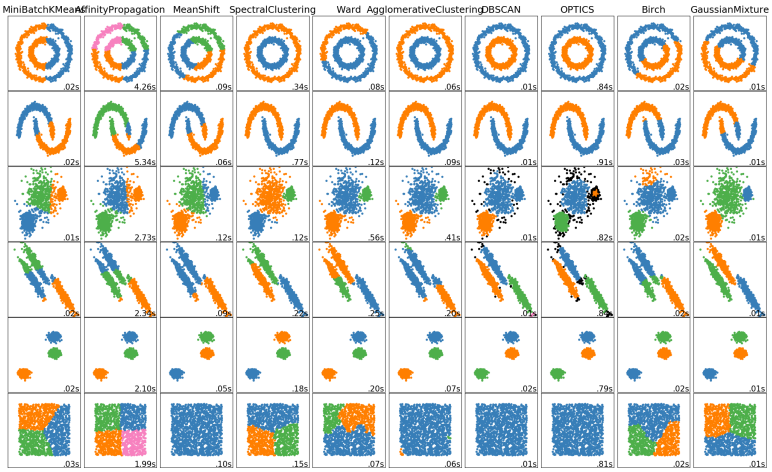
Dans la pratique, on peut dire que le clustering est un art - ou inversement un des domaines les plus piégeux de l'apprentissage statistique. Les illustrations sur les fruits montrent que plusieurs partitionement sont possibles selon l'objectif : en fonction de la taille, de la couleur, de la forme ... On peut bien sûr mélanger toutes ses caractéristiques, mais les différents résultats obtenus sont tous valides : nous n'avons pas d'évaluation non biaisée du résultat (à moins d'une connaisance extérieure au problème). Tout est possible et tout dépend de la mesure de similarité que l'on va utiliser.

Pour bien se rendre compte du problème, regardez la figure slide 5 et les différentes solutions de clustering possibles en fonction du nombre de clusters : 2, 3, 4 et 5 clusters. Quelle solution vous semble la meilleure ? On a tendance à préferer la solution à 3 ou 4 clusters (d'après les réponses de vos camarades des années précédentes). Le cerveau humain est conditioné à trouver des motifs dans tout ce qu'il observe. Mais en fait, la figure correspond à un tirage aléatoire dans le plan ... Autrement dit, aucun des clusters trouvés n'a de réelle signification.

2 Formalisation

Deux types de clustering sont en général possible, le *hard* où tout exemple n'appartient qu'à un seul cluster et le *soft* où l'on mesure un degré d'appartenance à chaque cluster (mixture de gaussiennes par exemple que vous avez déjà vu en MAPSI). Pour définir la similarité entre exemples,

on peut se baser sur une distance dans l'espace entre exemples (approches métriques), des similarités sur les exemples eux-mêmes non métriques (ne reposant pas sur une description dans $\mathbb{R}^n$ mais directement entre exemples) mais également sur la connectivité entre exemple sur des relations de localités, sur des modèles génératifs (type mixture de gaussiennes, Hidden Markov Model . . .). L'image ci-dessous (tirée de la documentation de sklearn, illustre différentes situations classiques de clustering et la différence de traitement entre les algorithmes.



Nous allons voir 3 approches représentatives dans ce cours : le clustering agglomératif, l'algorithme des k-moyenneset le clustering spectral (vous connaissez par ailleurs déjà la mixture de gaussiennes).

3 Algorithme agglomératif

Le principe des algorithmes agglomératifs est de construire un arbre de clustering : les feuilles sont des exemples et chaque nœud indique le regroupement en un cluster des nœuds fils. Le slide 11 présente sur la figure de gauche un exemple de co-clustering (clustering à la fois des colonnes et des lignes) et à droite un clustering hiérarchique. Dans ce type d'algorithme, la racine correspond à tous les exemples et en descendant dans l'arbre on obtient de plus en plus de clusters. La construction de la hiérarchie se fait d'habitude du bas vers le haut (bottom-up) : chaque exemple est mis dans une feuille distincte ; on touve ensuite les deux nœuds les plus similaires et un père est ajouté à ces deux nœuds ; ceux-ci sont enlevés de l'ensemble et le noveau nœud père est ajouté ; on réitère la fusion des nœuds les plus similaires jusqu'à n'avoir plus qu'un seul nœud. Pour juger de la similarité entre deux nœuds plusieurs mesures sont possibles : soit le barycentre des exemples qu'il contient est construit (méthode du représentant) et ce barycentre devient la description du nœud ; soit on garde l'information de tous les exemples qu'il contient et on peut choisir une pseudo-distance entre deux nœuds telle celles données en slide 10 : la distance entre deux clusters peut être le minimum des distances entre les exemples des deux nœuds, le maximum, ou la moyenne. D'autres variantes sont possibles. Un avantage de cet algorithme est qu'il ne nécessite pas forcément une description dans $\mathbb{R}^d$ des exemples, on peut utiliser une similarité entre exemples. Par ailleurs, le choix du nombre de clusters n'est pas fixé a priori mais peut être réglé et changé sans réappliquer l'algorithme (il suffit de changer de niveau dans l'arbre construit). Cependant, c'est un algorithme glouton : les solutions trouvées ne sont pas forcément optimales.

4 K-moyenne

Vu en MAPSI. Il faut retenir que c'est une application de l'algorithme E/M à un cas de hard clustering. C'est également un algorithme très utilisé.

5 Interlude gaussienne multivariée

Cette section est un rappel sur la signification des paramètres d'une gaussienne multivariée. En 1D, la constante $\frac{1}{(2\pi\sigma^2)^{0.5}}$ est bien sûr une constante de normalisation pour que l'intégrale de la densité soit égale à 1. En multivariée (gaussienne sur plusieurs dimensions), la moyenne μ se comprend simplement : il s'agit des coordonnées du centre de la gaussienne. Mais que signifie la matrice de covariance Σ à la place de la variance σ en 1D ? La suite essaye de vous donner l'intuition de cette notion.

Un cas simple en 2D est lorsque la matrice Σ est diagonal. Cela correspond en fait à 2 gaussiennes 1D indépendantes. Par l'application du théorème de Bayes, on retrouve la formule de la gaussienne 2D. On peut généraliser à plus grande dimension : une gaussienne multivariée de matrice de covariance diagonale correspond au tirage de manière indépendante sur chaque dimension i selon la gaussienne 1D centrée en μ_i et de variance σ_i^2.

Pour comprendre le cas générique où Σ n'est pas diagonale, il est plus simple de partir du cas d'une gaussienne 2D centrée-normée. Si on lui applique une transformation affine, son centre est translaté de la translation effectué μ. La variance elle reste à définir grâce à la notion de covariance qui indique pour chaque couple de dimensions comment l'une varie avec l'autre. En déroulant les calculs, on fait le lien entre la densité selon x et la densité de sa transformation affine y en faisant intervenir cette matrice de covariance. Une matrice de covariance peut toujours être diagonalisée en un produit UD^2U' avec UD la transformation affine sous-jacente. Ainsi une gaussienne multivariée générique n'est qu'une transformation (rotation, homothétie, ...) d'une gaussienne multivariée de matrice de covariance diagonale où D représente la variance sur chauqe dimension.

6 Clustering spectral

Le problème des algorithmes de clustering qui travaillent avec des représentants dans le plan (comme k-means) est qu'ils ne peuvent trouver que de clusters dépendant de la métrique utilisée (dans le cas de la distance euclidienne des clusters sphériques). Lorsque l'on veut tenir compte des relations de voisinages, d'autres algorithmes doivent être utilisés. Le clustering spectral est une approche intéressante utilisant le graphe de similarité entre les exemples pour inférer des clusters. Un graphe de similarité est construit à partir des exemples : chaque nœud du graphe est un exemple et deux nœuds sont reliés par une arête pondérée par la similarité entre les deux exemples associés (pas d'arête si les deux exemples ont une similarité en dessous d'un seuil fixé). L'objectif est alors de trouver comment partitioner le graphe en deux graphes connexes indépendants en supprimant le moins d'arêtes possibles, de manière à garder dans chaque composante connexe les exemples les plus similaires entre eux. Cela correspond au problème classique de coupe minimale (minimum cut). Le coût d'une coupe est la somme des poids sur les arêtes supprimées (que l'on peut normaliser afin d'équilibrer en nœuds les deux sous-graphes obtenus). C'est un problème NP-difficile que l'on va résoudre par relaxation.

On considère pour cela une matfice très utilisée dans les problèmes de graphe, la matrice laplacienne : elle correspond à $L = D - W$, D étant la matrice diagonale représentant les degrés des nœuds (la somme des poids des arêtes d'un nœud) et W la matrice d'adjacence. On considère souvent sa formulation normalisée : $D^{-1/2}LD^{1/2} = I - D^{-1/2}WD^{1/2}$. Toutes ses valeurs propres sont positives (rappel : les valeurs propres λ_i d'une matrice associées aux vecteurs propres v_i sont telles que $Lv_i = \lambda_i v_i$). Soit f un vecteur dans $\{-1, 1\}$ de taille n le nombre de nœuds du graphe représentant une partionement en 2 partitions des nœuds du graphe, alors $f'Lf = \frac{1}{2}w_{i,j}(f_i - f_j)^2$ (avec w_{ij} le poids entre les nœud i et j). Cela représente exactement le coût de faire une coupe selon ce partitionement : si $f_i = f_j$, les deux nœuds est dans la même partition, le poids n'est pas compté ; sinon le poids est compté. On peut généraliser à la version normalisée du Laplacien. On montre d'après la formule précédente que $f'Lf = 0$ pour $f = 1$, donc que la plus petite valeur propre de L est 0 (et on peut montrer que la multiplicité de cette valeur propre correspond au nombre de composantes connexes du graphe). On montre par ailleurs que $f'.1 = 0$ et que $\|f\| = 1$ (cette dernière condition correspond au fait qu'il n'y a pas de nœud non sélectionné dans f). L'objectif est donc de trouver le minimum de $f'Lf$ tel que $f'.1 = 0$ et que $\|f\|^2 = 1$. En relachant la contrainte de $f_i \in \{-1, 1\}$ et en autorisant les valeurs réelles, on peut montrer que la solution est le vecteur propre correspondant à la plus petite valeur propre de L non nulle. Un partitionement en plus de 2 parties est obtenue en étudiant les autres vecteurs propres.

7 Réduction de dimension

Le problème de réduction de dimension peut se résumer à trouver une transformation ϕ de l'espace des exemples $\mathbb{R}^d$ à un espace $\mathbb{R}^{d'}$ avec d' très inférieure à d tel que $\phi(x) \simeq x$, la transformation soit très similaire à la donnée d'entrée. Il y a de multiples raisons de s'intéresser à ce problème : réduction du bruit, visualiser les données (pour $d' = 2$), trouver l'information latente des données. Cela peut sembler paradoxale vu que dans les cours précédents notre problème était plutôt d'augmenter le nombre de dimensions pour améliorer l'expressivité. Mais pousser cette augmentation trop loin rend toujours un problème linéairement séparable et on se retrouve très facilement en sur-apprentissage. Réduire le nombre de dimensions permet d'éliminer les corrélations entre dimension et le bruit. Une première approche que vous avez du voir par ailleurs dans d'autres cours est l'analyse en composante principale : elle permet de décomposer (linéairement) dans une nouvelle base les exemples de manière à ce que les premières dimensions expliquent au mieux la variance des données : le premier axe principal est construit de telle manière à maximiser la variance (et donc de capturer le plus d'information) et itérativement les autres axes sont construits de la même manière.

Une autre technique très en vogue ces dernières années est l'apprentissage de dictionnaire. Le problème principal de l'ACP est que la nouvelle base est orthogonale : pas de redondance entre les différentes dimensions. Or il peut être utile d'autoriser de la redondance entre les dimensions de manière à mieux expliquer un certain nombre de motifs. Pour schématiser, si on pense à la décomposition d'un visage par exemple, plutôt que d'avoir un seul axe pour les yeux, un axe pour la bouche etc, on peut vouloir avoir plusieurs axes pour pouvoir mieux exprimer certaines caractéristiques des yeux et des bouches. Dans les approches par dictionnaire, on constitue une nouvelle base appelée dictionnaire (et chaque élément un atome) qui peut être très redondante. Une première approche est le compress sensing : on s'autorise un dictionnaire plutôt très large, mais pour la reconstruction de chaque donnée en combinant linéairement différents atomes, on ne s'autorise que très peu d'atomes. Cette approche a donné de très bons résultats en image pour le débruitage en particulier et la complétion.

Une autre approche très courante est la factorisation matricielle non négative : on ne s'autorise à combiner les atomes que avec des poids positifs. Cela favorise énormément l'interprétation de la décomposition obtenue. Grossièrement, on peut penser chaque atome comme un calque que l'on surperpose l'un par dessus l'autre pour obtenir la représentation finale. Le nom de factorisation matricielle vient de la formalisation : on recherche une matrice D d'atomes et une matrice de poids W toutes les deux positives telles que $X \simeq WD$. Cette approche est très utilisée en recommandation et quand il s'agit de faire du profiling (et dans beaucoup d'autres applications comme la séparation de source en traitement audio par exemple).

Nous terminons ce cours avec une petite transition vers le M2 en parlant des auto-encodeurs. Ce sont un type particulier de réseaux de neurones pour faire de la réduction de dimension. Le principe est d'avoir des couches larges avec beaucoup de neurones au début qui se rétrecissent petit à petit. La couche du milieu est la plus petite couche contenant un nombre de neurones égal à la dimension de réduction. Puis de manière symétrique, les couches s'élargissent pour aboutir à la dernière couche de la même taille que l'entrée. On entrîne ce réseau de tel manière que la sortie soit la plus similaire à l'entrée. Le réseau apprend ainsi à compresser l'information : la première partie du réseau s'appelle l'encodeur, la deuxième partie le décodeur. Le résultat de l'encodeur, avec un très faible nombre de neurones, est capable de capturer toute l'information nécessaire au décodage. En étudiant cette couche intermédiaire, on peut projetter les données dans l'espace 2D (si le nombre de neurones est de 2) ou s'en servir comme entrée en apprentissage. Ces réseaux permettent également de réduire le bruit. La forme variationelle que vous étudierez l'année prochaine est très utilisée pour la génération d'image entre autre.

Ressources :

— Pattern Classification de Duda, Hart, Stork. Trouvable en ebook.
— Le livre de Bengio et le site associé.