Telecom ParisTech

PAF

Contraste et Catégorisation

Antoine Bellami
Aurélien Blicq
Clément Bonet
Benoit Malézieux
Louis Penet de Monterno
Bastien Vagne

29 juin 2018

Sommaire

1 Introduction

Les techniques de clustering produisent des classes d'objets et des jugements d'appartenance à ces classes. L'idée du contraste consiste à opérer une différence vectorielle entre un objet et la classe la plus proche, puis de caractériser cette différence. Ainsi, une tomate noire sera décrite comme telle parce que la différence avec le prototype "tomate" sera proche du prototype "noir".

L'opération de contraste permet à une IA :

- de comprendre le sens de "petite bactérie" et de "petite galaxie" sans passer par l'ensemble des objets petits, car dans les deux cas "petit" correspond au contraste avec le prototype ;
- de produire des descriptions pertinentes : "c'est un chanteur qui a dix millions de vues sur Youtube", car c'est ce qui le distingue du prototype de chanteur ;
- de détecter des anomalies : un chat qui parle ;
- d'apprendre à partir d'un seul exemple : un enfant qui voit pour la première fois un chat blanc, noir aux extrémités, comprend et mémorise l'expression "chat siamois."

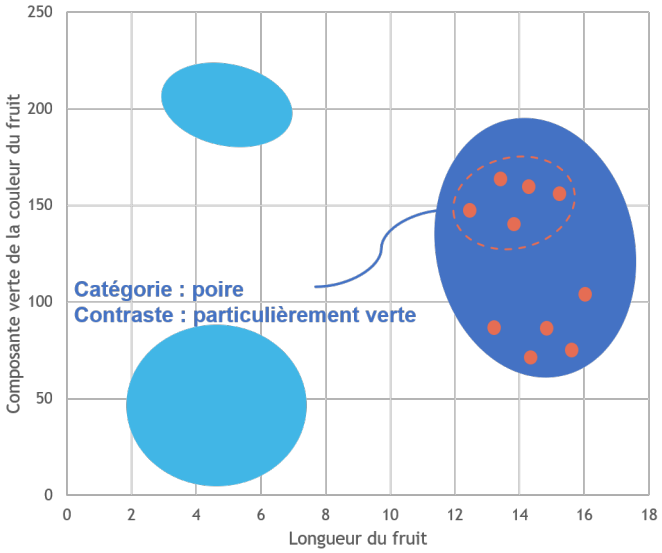

FIGURE 1 – Exemple d'utilisation du contraste

Ce projet a pour objectif la réalisation [1] d'une première implémentation du contraste, que l'on teste sur des jeux de données réel ou artificiel. L'opération de catégorisation sera appliquée à deux reprises : une fois sur les objets eux-mêmes, une deuxième fois sur les contrastes (différences objets-prototypes). La deuxième application du contraste est censée faire ressortir des modifieurs interprétables comme des adjectifs. On veut finalement décrire de façon pertinente un nouvel objet donné. La suite du projet consiste à placer le contraste au coeur de l'activité d'apprentissage, en l'utilisant pour former les catégories. L'idée est de permettre un apprentissage pertinent en présence de (très) peu de données.

1. `https://github.com/Amocokadys/PAF_contrastes_categorisation/tree/master`

Ce problème pourrait avoir plusieurs applications. Prenons l'exemple d'un système de machine learning devant reconnaître des images. Ce système, pour pouvoir fonctionner correctement, nécessite des quantités gigantesques de données, parfois difficiles à collecter. Les techniques de machine learning sont actuellement extrêmement désemparées lorsqu'elles ont affaire à peu de données. L'apprentissage incrémental permettrait de reconnaître des motifs en "one-shot" , c'est-à-dire avec une ou quelques données d'apprentissage. Notre objectif est d'implémenter ce type d'apprentissage, pour des données plus simples à traiter que des images, en utilisant le concept de contraste afin de construire les clusters. Une autre application du contraste serait le repérage d'exceptions.

2 Méthode Offline

Cette section présente l'architecture11 de la méthode que l'on a appliqué afin d'effectuer le contraste et de le tester.

Tour d'abord, on travaille dans un espace dont la dimension est le nombre d'attributs qui nous intéressent (la taille, la forme et le goût de fruits par exemple).

Dans un premier temps, il faut catégoriser les données afin de les classifier par cluster. Pour cela, on peut utiliser divers algorithmes tels que k-means ou encore GMM (Gaussian Mixture Model). On a commencé par effectuer une classification grâce à l'algorithme de k-means. Il existe différentes possibilités pour choisir le k, c'est-à-dire le bon nombre de catégories : on peut notamment utiliser une métrique particulière pour comparer les résultats de l'algorithme avec différentes valeurs. Le problème est qu'il n'existe pas un nombre parfait de clustering [1], il faut donc essayer de choisir ce nombre de manière la plus optimale possible.

L'algorithme k-means est bien illustré par les diagrammes suivants :

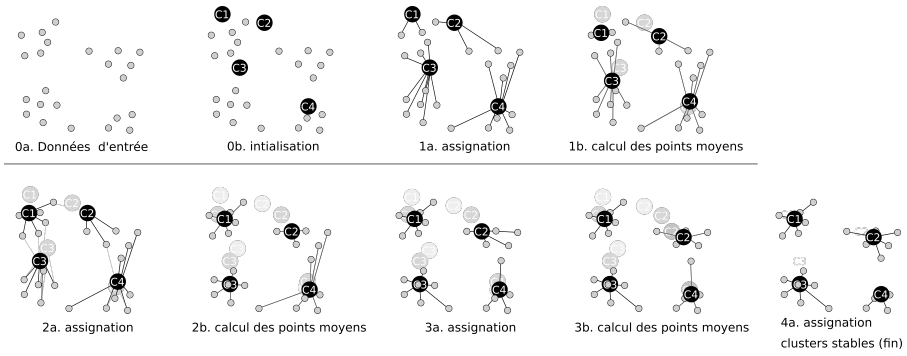

FIGURE 2 – Illustration du déroulement de l'algorithme des k-means - Mquantin (Wikipedia)

Un cluster est défini par un centre et un ensemble de points (vecteur centre et vecteur écarts-types). Pour une meilleure classification, on peut décider de normaliser par rapport aux écarts-types sur chacune des dimensions (grâce à la matrice de covariance).

Pour contraster chaque cluster, on calcule la différence entre chaque point et le centre afin de se ramener autour de l'origine. On catégorise donc les différences en déterminant la proximité de chaque point avec le centre (les points éloignés du centre ont un contraste élevé avec la moyenne des points, ils auront des caractéristiques particulières par rapport à leur catégorie).

On commence par travailler sur un jeu de données simple (qui comporte au moins une dizaine de dimensions) et que l'on maîtrise bien : les fruits. Après avoir rentré à la main quelques données de fruits, on construit la base de données en bruitant les différentes caractéristiques de manière anisotrope : on ne bruite pas tout en même temps de la même manière pour espérer avoir des groupes de contraste interprétables.

En parallèle, on essaye de créer un jeu de données plus conséquent avec plus de dimension. Pour cela, on utilise une base de données de livres open-source en français dans lesquels on va extraire de manière pertinente les mots les plus fréquents grâce à l'algorithme TF-IDF [2]. Ces mots correspondront à nos dimensions. Finalement, on a pas utilisé cette base de données pour nos résultats car les résultats n'étaient pas assez interprétables.

3 Développement - Méthode Offline

3.1 Premier programme

3.1.1 Clustering

On utilise l'implémentation de k-means fournie par le module python sklearn. La difficulté d'application de l'algorithme consiste à trouver le nombre optimal de clusters, c'est-à-dire le nombre de classes dans lequel on va partager notre ensemble de données.

Pour cela, on a commencé par appliquer la méthode du "coude" (elbow method) qui consiste à appliquer l'algorithme pour plusieurs k différents et à tracer les sommes des variances de la distance euclidienne de chaque cluster par rapport à leurs centres. On repère ensuite la valeur de k "qui se situe dans le coude". Cette valeur de k sera le nombre de classes optimales.

On a trouvé cette méthode plutôt graphique peu précise informatiquement. Il a donc été décidé d'essayer d'appliquer une métrique "silhouette "[3]. Celle-ci consiste à calculer la différence entre la distance moyenne d'un point x avec tous les points de son cluster $a(x)$, et la plus petite valeur que pourrait prendre a(x) si x appartenait à un autre cluster $b(x)$. On la normalise par le max afin d'avoir une valeur entre -1 et 1 :

$$s(x) = \frac{b(x) - a(x)}{max(a(x), b(x))}$$

où

$$a(x) = \frac{1}{|C_k| - 1} \sum_{u \in C_k, u \neq x} d(u, x)$$

et

$$b(x) = \min_{l \neq k} \frac{1}{|C_l|} \sum_{u \in C_l} d(u, x)$$

Il faut donc calculer cette métrique pour tous les points et prendre la moyenne. Pour un point, plus la métrique est proche de 1 et plus l'assignation du point à son cluster est satisfaisante. On en déduit donc que plus la silhouette sera proche de 1, et plus le clustering sera bon.

Finalement, on s'est rendu compte que la méthode silhouette était strictement croissante pour beaucoup trop de k et comportait donc peu d'intérêt pour nous. Du coup, on s'est rabattu sur la méthode du coude en utilisant comme critère les pentes afin de chercher à trouver le k présent dans le coude. Comme les points ne forment pas toujours un "coude" parfait et ne sont pas forcément décroissants, on a utilisé une régression polynômiale d'un ordre assez élevé afin d'approximer la courbe. On choisit alors un critère comme par exemple avoir une pente inférieur à 10% de la première pente. Par exemple sur la figure suivante 3, le coude se trouverait vers k=10 qui serait alors le nombre de clusters optimal. De plus, on effectue une régression polynômiale avant d'appliquer le critère de pente afin d'éviter des points marginaux qui donnerait un nombre de classes pas forcément optimal.

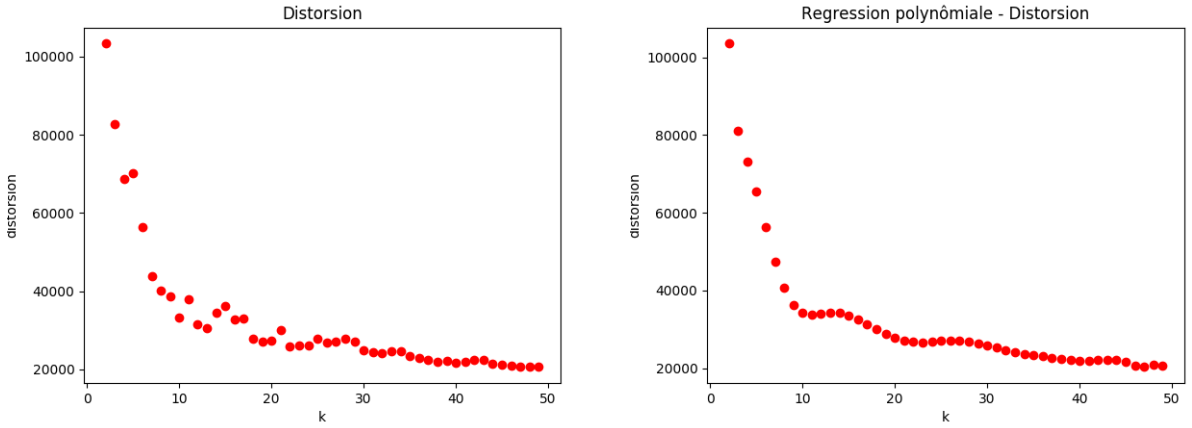

FIGURE 3 – Illustration de la méthode du coude

3.1.2 Covariance et contraste

- Après avoir appliqué le premier k-means sur les données, et avoir eu un premier clustering, on ne garde qu'un certain pourcentage (à choisir) des points les plus proches de chaque centre de cluster dans le but d'éliminer les points mal classifiés.
- On calcule ensuite les matrices de covariance des variables aléatoires sur l'ensemble des points de chaque cluster. On définit la distance d'un point à un cluster de la sorte : soit C^{-1} l'inverse de la matrice de covariance du cluster et A son centre. $d = \|C^{-1}(M - A)\|$ est la distance normalisée du point M au cluster. Pour chaque point qui n'est pas encore dans un cluster, on cherche quel est le cluster le plus proche selon ces distances et on l'associe à notre point, sans modifier la matrice de covariance du cluster afin de séparer la phase d'apprentissage des clusters de la phase d'attribution des clusters à un objet. On utilise ainsi la distance en nombre d'écarts-types.

  Démonstration

  Le but de cette démonstration est d'expliquer pourquoi la multiplication par l'inverse de la matrice de covariance permet d'obtenir une matrice normalisée par l'écart-type sur les axes décorrelés.
  On note C la matrice de covariance du vecteur aléatoire $(X_i)_{i\in[\![1;n]\!]}$. C est symétrique donc d'après le théorème spectral, C est diagonalisable en base orthonormée, et ses valeurs propres sont les écarts-types σ_i. Donc $C = PDP^{-1}$ où $D = (\sigma_i)_{i\in[\![1;n]\!]}$. On note M un point à normaliser et M' la matrice correspondant à ce point dans la bonne base de normalisation (en changeant de repère orthonormal).

$$\|C^{-1}M\| = \|P^{-1}D^{-1}PM\| = \|D^{-1}M'\| = \sum \frac{1}{{\sigma_i}^2} {M'_i}^2$$

- Pour effectuer le contraste, on calcule la différence de chaque donnée avec le centre du cluster.
- On effectue une projection non linéaire appelée sharpening (provisoirement, on ne garde que les 10% des plus grandes composantes de chaque vecteur).
- On applique l'algorithme de k-means puis les deux premières étapes de ce paragraphe à ces données.

Résultat

Les clusters ne sont pas vraiment réalistes. Le problème vient de la faible proportion de points gardés par cluster(20%) et du fait que les gros clusters ont tendance à absorber tous les points en raison de leur écart-type

supérieur. On tentait d'utiliser un critère pertinent pour un seul cluster afin de comparer les éléments de deux clusters distincts. Or si un cluster déterminé par k-means est beaucoup plus diffus que les autres, on constate expérimentalement qu'il absorbe énormément de points lors de la phase de reconstruction avec le contraste.

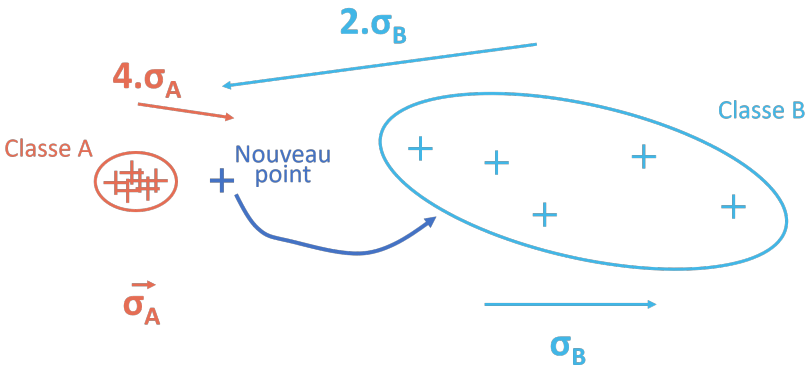

FIGURE 4 – Illustration du problème de variance

3.2 Second programme

L'idée d'utiliser la variance pour normaliser les distances n'ayant pas abouti, nous avons décidé de sauter l'étape de sélection et de faire confiance à l'algorithme de clustering. Après quelques hésitations, on s'est rendu compte que l'algorithme de k-means avait tendance a renvoyé des clusters de forme plutôt "sphérique". En effe, le critère est la distance euclidienne des points avec le centre ce qui n'était pas forcément pertinent pour nos jeux de données. On a donc aussi testé d'appliquer l'algorithme de GMM à la place. Cet algorithme approxime les variables aléatoires par des sommes de gaussiennes ce qui fait qu'au final, les clusters auront des formes plutôt "gaussiens". GMM nécessite aussi de connaître le nombre de clusters, ou alors de l'estimer avec la méthode du coude par exemple.

Pour le contraste, nous avons dans un premier temps envisagé de ne garder que la dimension la plus particulière, c'est-à-dire la dimension telle que le contraste soit maximal. L'utilisation du sharpening avec un seuil à paramétrer interviendra dans un second temps.

Pour tester notre programme, il est nécessaire de pouvoir vérifier manuellement que les clusters de catégorie et de contraste sont cohérents et que les éléments extérieurs sont rattachés aux bons clusters. Pour cela, nous avons crée une base de données avec des labels caractérisant les fruits. Par exemple, un fruit du type abricot et avec une grande longueur, une petite largeur et un coefficient rouge élevé sera qualifié de : abricot grand fin rouge. On assigne un nom à chaque cluster afin de vérifier la cohérence de nos résultats. Pour déterminer l'étiquette qui sera accolée à un cluster, on utilise le label le plus représenté au sein de ce cluster (anchoring).

Résultat

Les résultats de l'algorithme étaient plutôt satisfaisants. Par exemple, lorsqu'on donnait en entrée un abricot avec une longueur aberrante, il renvoyait la bonne catégorie (ici abricot) ainsi que le label "longueur" qui était atypique.

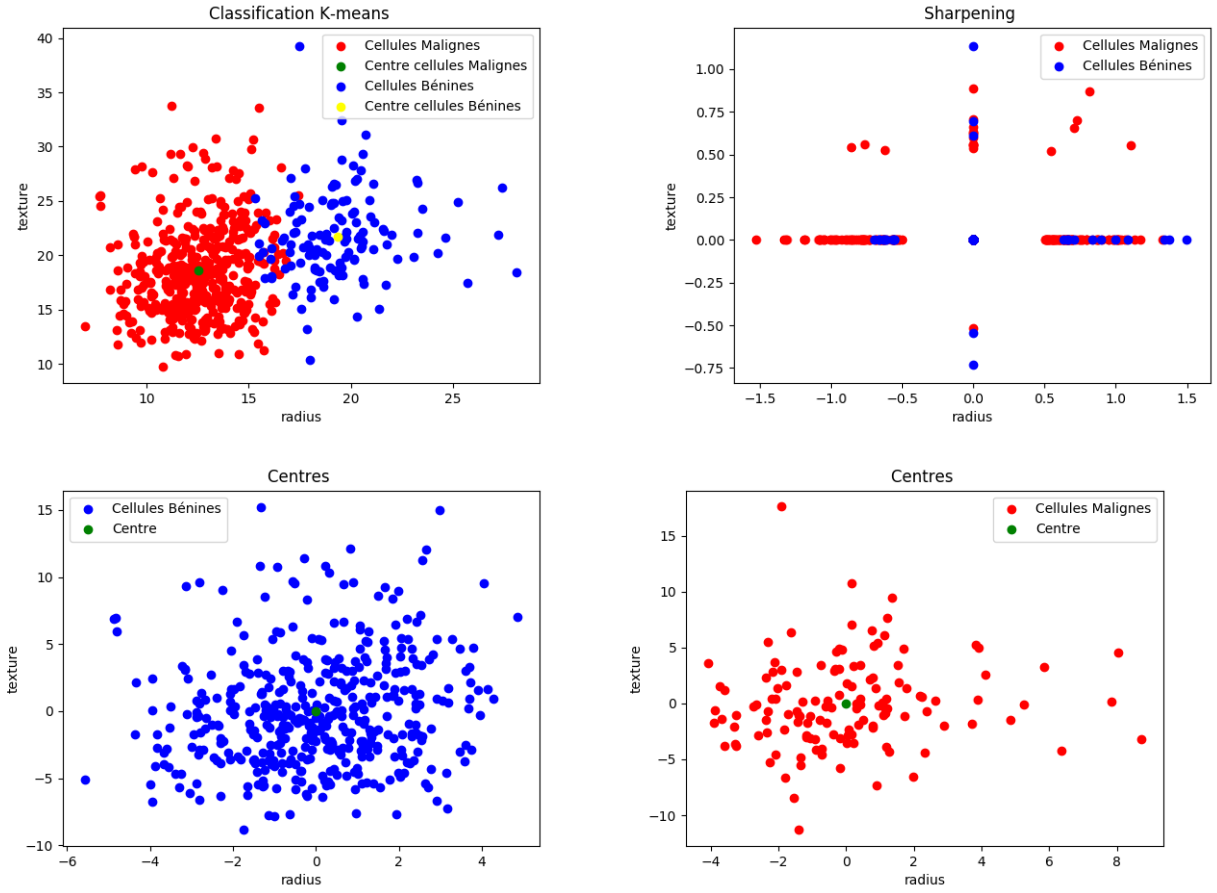

FIGURE 5 – Exemples de clusters et de sharpening obtenus

3.3 Troisième programme

Dans un second temps, nous avons appliqué le second clustering à tous les vecteurs de contraste en appliquant le sharpening à hauteur de 10% du contraste maximal. Plus précisément, le programme retourne les caractéristiques particulières trouvées grâce au contraste.

La figure suivante illustre les résultats que l'on obtient avec notre jeu de données de fruits :

Voici les résultats obtenus par la catégorisation sur notre base de données artificielle de fruits.

Voici les résultats obtenus par l'opération de contraste sur la même base de données de fruits.

Extrait d'un jeu de données de fruits décrits :

Longueur	Contraste longueur	Catégorie du fruit	Catégorie du contraste
5.1	+ 1.2	Fraise	Grand
12.7	- 1.7	Poire	Petit

Entrées partielles

Sorties

FIGURE 6 – Principe

Catégories	« tomate »	« banane »	« cerise »	« pomme »	« abricot »	« patate »	« prune »	« prune »	« poire »	« prune »
Nombre d'éléments	100	100	201	100	200	100	27	35	100	37
Eléments bien placés	100 %	100 %	49,75 %	100 %	50 %	100 %	100 %	100 %	100 %	100 %
Eléments mal placés			Poivrons : 49,75 % Prunes : 0,5 %		Carottes : 50 %					

FIGURE 7 – Résultats de la catégorisation

Label	Centre (en nombre d'écarts-types) (les valeurs non spécifiées sont nulles)	Interprétation	Pourcentage
['r+']	R = 2,35	Rouge	6,12 %
[]		normal	19,45 %
['longueur+', 'largeur+', 'b+', 'sucre-', 'eau+', 'fibres-']	Longueur = 0,89, largeur = 1,36, b = 1, 42, sucre = -2,25, eau = 0,71, fibres = -1,83	aucune	2,19 %
[]		normal	21,10 %
['longueur+']	Longueur = 1,28	long	2,18 %
['largeur+']	Largeur = 1,63	large	4,5 %
['longueur+', 'largeur+]	Longueur = 0,93, largeur = 1,06	gros	5,89 %
['v-', 'b+', 'eau+', 'sucre+']	V = -0,63, b = 1,45, eau = 1,05, sucre = 2,02	aucune	1,64 %
['r-', 'b-', 'v-']	R = -0,83, b = -0,76, v = -0,62	Clair	6,2 %
['longueur+', 'eau-', 'sucre+', fibres-]	Longueur = 0,72, eau = -1,23, sucre = 1,42, fibres = -1,06	aucune	5,28 %
['sucre-']	Sucre = -2,31	Peu sucré	8,31 %
['eau-', 'sucre+', fibres-]	Eau = -0,60, sucre = 1,33, fibres = -1,42	aucune	4,53 %
['r-', 'v+', 'b+']	R = -0,62, v = 0,88, b = 1,04	jaune	8,42 %
['longueur+', 'largeur-']	Longueur = 1,58, largeur = -1,38	Long et fin	3,89 %

FIGURE 8 – Résultats du contraste

Nous avons développé une interface graphique pour visualiser plus facilement les résultats. L'interface permet d'entrer des valeurs sur chaque dimension puis de connnaître la catégorie et le contraste de l'objet décrit. De plus, le programme affiche un graphe avec deux dimensions choisies en abscisse et en ordonnée.

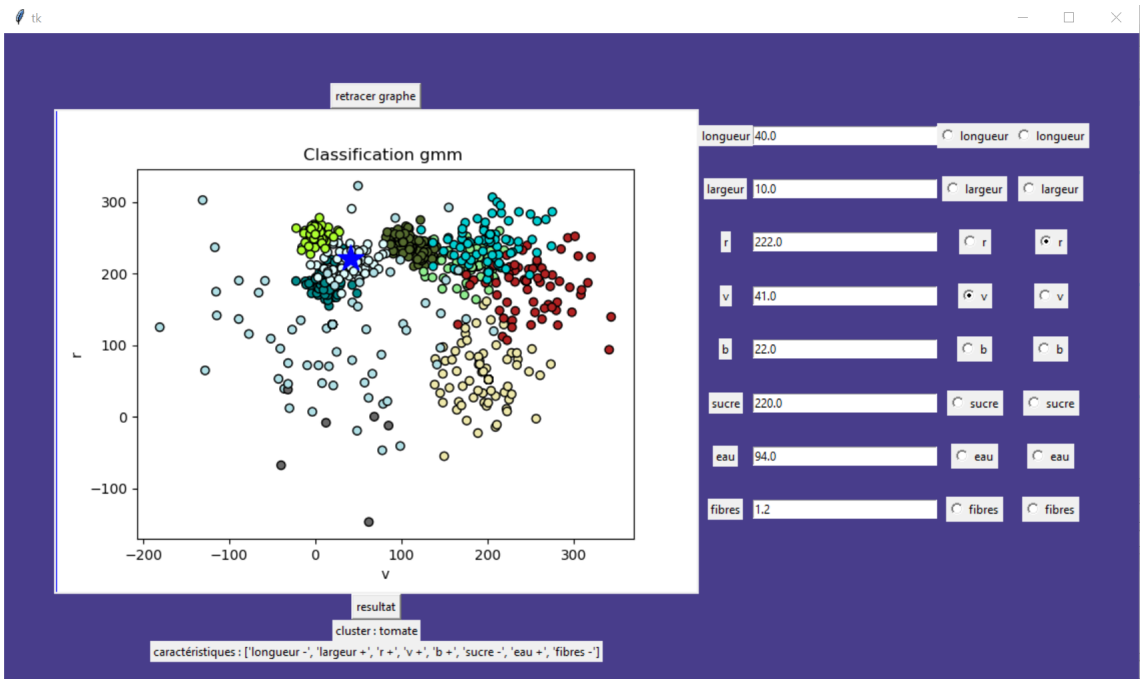

FIGURE 9 – Interface graphique

3.4 Quatrième programme

Dans cette version du programme, on va gérer les infinis.

Tout d'abord, on fait une étape alternative entre le GMM et le contraste. On cherche à garder les points assez proches du centre du cluster. Pour cela, on fixe par un seuil que l'on nommera infini et qui sera par exemple de 3 écarts-types. On crée un nouveau cluster appelé "core" qui contiendra les points les plus au centre du cluster.

On définit une nouvelle donnée appelée "prototype" qui va contenir les centres des core ainsi que les écarts-types selon chaque dimension.

Dans la partie contraste, nous avons décidé d'effectuer le sharpening par ligne. En effet, les valeurs étant normalisées, cela permet de ne garder que les dimensions les plus importantes de chaque objet. De plus, on effectue ce contraste sur les points du core.

Pour la normalisation, on peut avoir des cas où les écarts-types sont nuls et donc la valeur infinie ce qui est problématique pour effectuer le second GMM sur le contraste. Lorsqu'un écart type est nul, on a alors deux cas :

- soit la valeur d'un objet sur cette dimension est nulle et alors la valeur normalisée doit aussi être nulle (le point a la même valeur que le centre sur cette dimension)
- soit la valeur d'un objet sur cette dimension est non nulle et alors la valeur normalisée va valoir l'infini. Il faut alors mettre cet objet à part.

4 Méthode incrémentale

Pour tenter de régler le problème des petits jeux de données qui ne permettent pas d'obtenir des clusters satisfaisants, l'idée de la méthode incrémentale est de créer ou de compléter les clusters au fur et à mesure de l'obtention des données. Nous nous inspirâmes de la biologie : de la même manière que les espèces animales et végétales peuvent être classifiées sous forme d'arbre, correspondant à l'évolution darwinienne, il paraît raisonnable que des données quelqonques puissent appartenir à une hiérarchie de catégories, qui forme un arbre. Cet arbre est formé de noeuds, qui correspondent à des catégories, imbriquées les unes dans les autres, et de feuilles, qui représetent des données. On construit nos catégories de manière itérative, de manière à ce que les clusters formés soient cohérents et permettent de classifier correctement quelques données.

Le groupe a eu l'idée d'utiliser des arbres pour ajouter les éléments au fur et à mesure. Chaque noeud de l'arbre correspond à un cluster, et ses sous-arbres fils sont les éléments appartenant à ce cluster.

L'ajout d'une nouvelle donnée à l'arbre se fait de manière récursive. Supposons que cette donnée corresponde à la description d'un chat, il faut d'abord déterminer si, parmis tous les êtres vivants, cette donnée est plutôt un vertébré, ou un invertébré. Une fois qu'il sera identifié comme vertébré, il faudra le classifier parmi les poissons, les oiseaux, etc. Ainsi, en partant de la racine, le chat subit une suite de classifications successives, qui le font descendre dans l'arbre, jusqu'à ce qu'il rejoigne les autres chats, dans un éventuel sous-cluster constitué de chats.

L'un des avantages de cette approche est la possibilité d'un apprentissage amnésique : les données les plus communes pourraient être oubliées, au profit des seuls clusters, tandis que les données plus exceptionnelles seraient conservées. Pour discriminer les données utiles à conserver, un calcul de complexité pourrait être utilisé. Celle-ci peut être estimée en sommant les complexités du chemin dans l'arborescence, et celles correspondant au rang de la donnée parmi les données soeurs.

Algorithme

Pour initialiser l'arbre, on considère une racine vide que l'on peut assimiler à la classe générale objet. Quand le premier élément arrive on l'ajoute au noeud. on procède de même avec le deuxième élément. A partir du troisième élément, on minimise une certaine valeur comme la distance euclidienne aux autres clusters ou la complexité de l'arbre pour choisir le noeud où placer l'objet, ou décider de créer un cluster. Quand un élément arrive, on le compare de manière récursive aux éléments de l'arbre. S'il est proche d'un cluster, on regarde le sous-arbre qui correspond à ce cluster. Sinon, on crée un nouveau cluster au point actuel.

Pour comparer les éléments entre eux, la première idée a été d'utiliser la distance euclidienne et la variance dans l'esprit de la première méthode. Cependant, il nous parût plus pertinent de nous inspirer de la théorie de la complexité, en disant qu'une donnée est proche d'une autre si elle a un grand nombre de dimensions en commum. Par exemple un tigre et un chat diffèrent par leur taille, mais sont identiques sur les dimensions "nombre de pattes", "nombre de griffes", etc.

Nous avons donc décidé d'utiliser la complexité de Kolmogorov [4]. A chaque fois qu'un élément doit être traité, on le place à l'endroit de l'arbre qui minimise la complexité à la racine. Cette complexité est calculée de manière récursive sur l'arbre en fonction de plusieurs cas : la création d'un cluster ou l'ajout de l'élément dans un sous-cluster. La complexité dépend de l'importance des clusters (en fonction de leur nombre d'éléments), de leur distance à l'élément à intégrer et de la complexité des sous-clusters.

Résultat

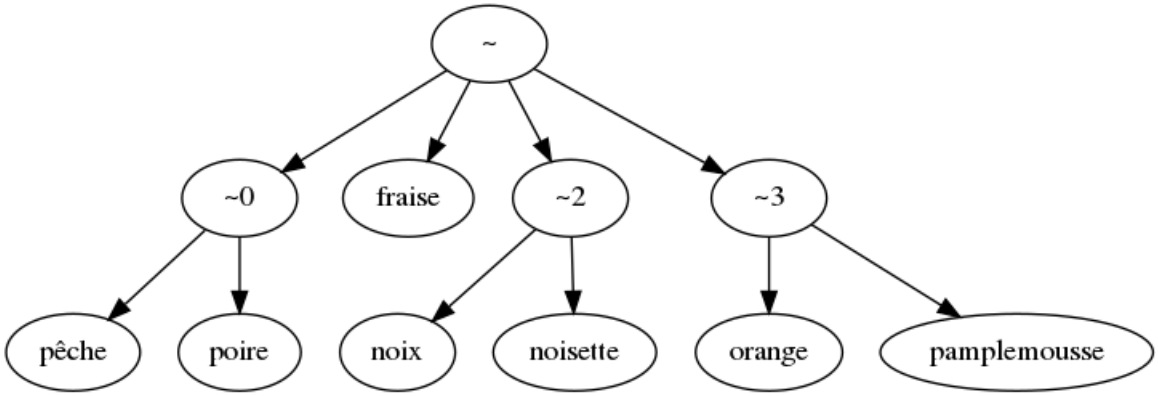

FIGURE 10 – Arbre renvoyé par l'algorithme incrémental

5 Perspectives

Plusieurs cas d'utilisation possible des méthodes citées peuvent être imaginés. Par exemple, il est envisageable de faire de la détection d'anomalies à partir du contraste. A partir d'un jeu de données quelconques contenant plusieurs catégories d'objets, il serait possible de repérer les caractéristiques atypiques d'un élément par rapport à sa catégorie.

On pourrait aussi imaginer d'utiliser ces méthodes pour faire du one-shot learning ou de l'apprentissage sur faible volume de données. Dans une première approche, ces méthodes permettraient d'avoir quelques résultats avec une petite quantité de données. Par exemple, il serait possible de créer les clusters et de les labelliser au fur et à mesure de l'arriver des données et de la classification incrémentale.

Références

[1] Jon Kleinberg. *An Impossibility Theorem for Clustering.* PhD thesis, Cornell University, 2002.

[2] Josiane Mothe and Faneva Ramiandrisoa. *Extraction automatique de termes-clés : Comparaison de méthodes non supervisées.* PhD thesis, Conférence en Recherche d'Informations et Applications, 2016.

[3] Maha Ghribi, Pascal Cuxac, Jean-Charles Lamirel, and Alain Lelu. *Mesures de qualité de clustering de documents : Prise en compte de la distribution des mots clés.* PhD thesis, CNRS, 2010.

[4] Bruno Durand and Alexander Zvonkin. Kolmogorov complexity. https ://www.labri.fr/perso/zvonkin/Research/kolmathan.pdf.

A Distribution des tâches

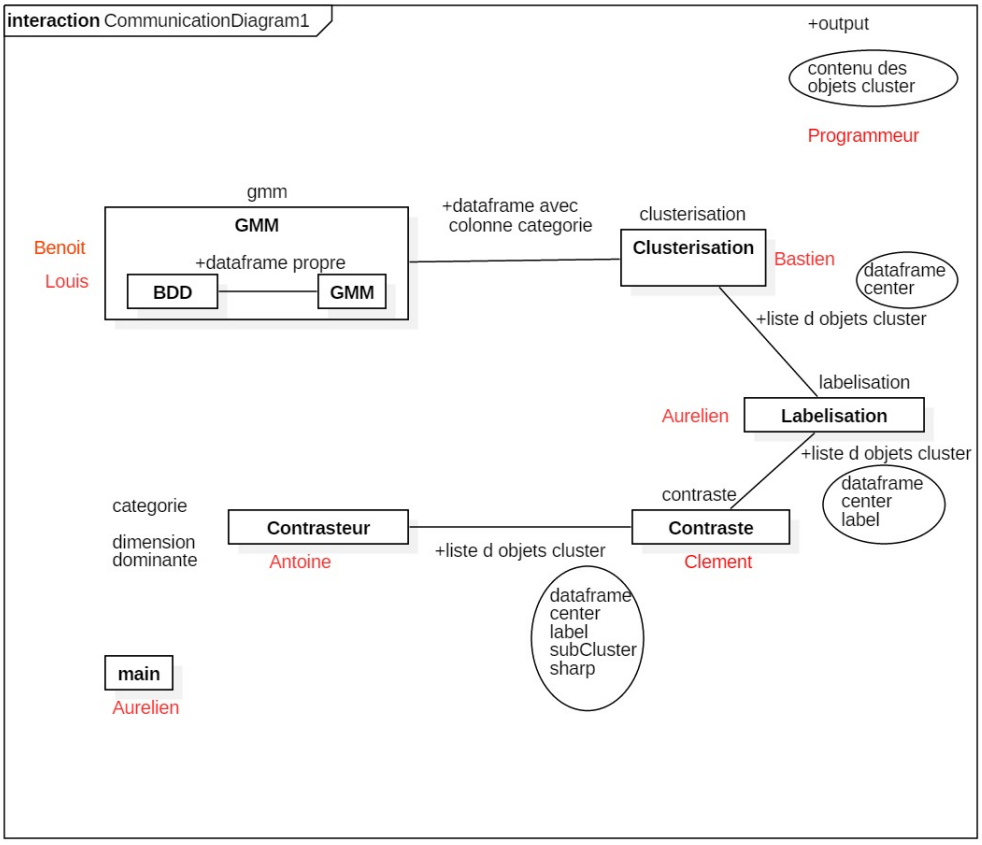

FIGURE 11 – Diagramme d'activité

Répartition des rôles au sein du groupe :

Méthode Offline : La gestion et le traitement des bases de données ont été confiés à Louis et Benoît. La première catégorisation avec utilisation de K-means et GMM a été codée par Benoît et Clément. La transformation en cluster avec traitement des labels a été faite par Aurélien et Bastien. Le traitement du contraste et l'intégration ont été gérés par Antoine, Aurélien et Clément.

Méthode incrémentale : Le clustering hiérarchique a été développé par Louis. La réflexion sur l'utilisation de la complexité a été faite par Louis, Bastien et Benoît.

Interface graphique : L'interface graphique a été développée par Bastien.

Rapport, présentation et poster : Le rapport a été écrit par Clément, Benoît et Antoine. Le poster a été imaginé par Antoine et Benoît. Les diapositives de présentation ont été pensées par Antoine et Benoît.

Qu'est-ce qu'un dataframe ? Un dataframe est un objet de la librairie pandas de python qui correspond à un tableau à double entrée avec des labels pour les lignes et les colonnes et qui est donc bien adapté à nos problèmes.