Science des données
IFT 3700

Travail Pratique 1

Partitionnement MNIST

David Raby-Pepin [918119]
Matiar Erfanian Azmoudeh [20031318]
Yifu Zhou (cours abandonné)
Hamza Bellk (cours abandonné)

Université de Montréal
Automne 2019

Introduction

L'objectif de ce travail pratique était de développer des mesures de similarité originales spécifiquement construites pour une utilisation sur l'ensemble de données MNIST et, en deuxième temps, comparer leur performance sur une gamme prédéfinie d'algorithmes de partitionnement et de réduction de dimensionnalité.
Les algorithmes considérés dans ce travail sont les suivants:

- k -médoïde
- Partition binaire
- PCoA
- Isomap
- KNN

Un comparaison de la performance de chaque algorithme implémenté avec différentes mesures de similarité a été effectuée. Dans notre cas, l'utilisation de la distance euclidienne a servi de mesure de base, indiquant le niveau de performance à battre à l'aide des mesures originales développées.

Notre équipe a développé deux mesures de similarité originales qui seront décrites et évaluées dans ce rapport:

- Buffered pixel mapping (BPM)
- Average nearest pixel (ANP)

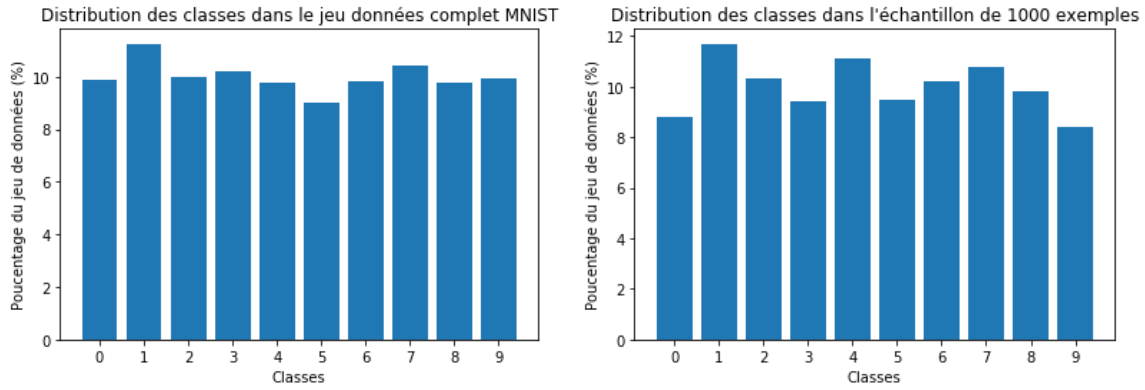
Outils utilisés
Langage de programmation: Python
Framework: Google Colab, Jupyter Notebook
Librairie: Scikit-learn

Méthodologie

Ensemble de données utilisé
Pour ce travail, le jeu de données MNIST de 70 000 exemples étiquetés contenant 784 caractéristiques a initialement été importé de la librairie scikit-learn. Malgré nos tentatives de rendre l'exécution des deux mesures de similarités originales développées la plus efficace possible, les temps d'exécution des algorithmes considérés implémentant nos mesures sont demeurés extrêmement longs. Nous avons donc été forcés d'effectuer les tests de comparaison de la performance des différentes mesures sur un échantillon très restreint du jeu de données MNIST. L'entièreté des tests présentés dans ce rapport repose sur l'utilisation d'un ensemble de 1 000 exemples échantillonnés de manière aléatoire à partir du jeu de données

MNIST. La taille limitée de l'échantillon employé engendre certains problèmes qui sont couverts plus bas.

Il est également important de noter que la distribution des classes (étiquettes) dans le jeu de données complet MNIST est légèrement non-uniforme. Étant donné que l'échantillon utilisé de 1 000 exemples tirés aléatoirement est trop petit pour refléter la distribution du jeu de données complet de manière satisfaisante, la distribution des classes dans l'échantillon est encore moins uniforme.



Problèmes rattachés à la taille de l'échantillon utilisé

Tel que mentionné, une des problèmes engendrés par l'utilisation d'un petit échantillon de 1 000 exemples est que la distribution des classes dans l'échantillon ne reflète pas exactement la distribution du jeu de données MNIST complet. Nous jugeons tout de même que la distribution observée pour l'échantillon est assez uniforme et similaire à celle du jeu complet compte tenu de l'utilisation escomptée. De plus, nous anticipons qu'une baisse de performance potentielle due à l'échantillonnage affecterait les différentes mesures de similarité de la même manière, leur comparaison demeurerait donc valide.

Un problème que nous considérons plus important est l'augmentation de la variance des modèles considérés (erreur d'estimation), soit la variabilité des résultats en fonction du jeu d'apprentissage fourni, due à l'utilisation d'un échantillon très petit. Une solution possible à ce problème serait de répéter les tests sur chaque modèles plusieurs fois pour toutes les mesures utilisées et de prendre la moyenne des résultats afin de comparer les performances des différentes mesures. Malheureusement, nous n'avions pas le temps de répéter la série de tests effectués à multiples reprises et sommes conscients des ramifications sur la validité de nos observations.

Évaluation de la performance

Pour les algorithmes de réduction de dimensionnalité, soit PCoA et Isomap, l'évaluation de la performance s'est effectuée de manière qualitative. Nous avons réduit la dimension des exemples fournis en entrée aux algorithmes de 784 à 2 caractéristiques et avons représentés les résultats sur un graphique à 2 axes. Nous avons ensuite observé les graphiques générés par l'utilisation des différentes mesures de similarité (ainsi que les diverses combinaisons de leurs hyperparamètres) afin d'identifier ceux démontrant les agglomérations les plus denses et les plus distinctes entre les différentes étiquettes.

Pour les algorithmes de partitionnement non-supervisés, soit K-médoïdes et la partition binaire, l'évaluation de la performance s'est effectuée à l'aide de la mesure V. Nous avons choisi d'utiliser un indice d'évaluation externe étant donné que nous avions accès aux étiquettes du jeu de données, constituant une partition cible qui représente la solution optimale désirée.

Pour le seul algorithme supervisé, soit KNN, nous avons opté pour la macro-moyenne du ‘’recall’’ afin d'évaluer la performance. Ce choix est rattaché à la taille limitée de l'échantillon du jeu de données MNIST utilisé. Étant donné que la distribution dans l'échantillon ne reflète pas la distribution du jeu de données complet de manière satisfaisante et que la distribution des classes sur le jeu de données complet est légèrement non-uniforme, elle est encore moins uniforme pour l'échantillon de 1 000 exemples. La macro-moyenne du ‘’recall’’ est donc utilisée afin d'accorder autant d'importance à la performance sur chaque classe, malgré leur distribution non-uniforme.

Mesures de similarité développées

Base commune

Les deux mesures développées effectuent un calcul de similarité sur une paire d'exemples reçue en entrée et la partie initiale de leur code est la même. Ce code se charge de:

- Transformer les exemples reçus en entrées de vecteurs de 784 dimensions en matrices de dimensions 28x28. Cette étape sert à faciliter les opérations subséquentes sur ces exemples.
- Faire une liste de l'emplacement de tous les pixels qui sont plus foncés qu'une valeur fournie en entrée (hyperparamètre nommé ‘’Binary threshold’’). Seulement les pixels dans cette liste seront considérés pour le restant du code. Les autres pixels sont considérés comme complètement blancs.
- Identifier la liste (parmi les deux listes) qui compte le plus de pixels foncés et lui assigner la première position (liste1). Cette étape sert à assurer que les

mesures soient pseudo-symétriques, une condition requise afin de pouvoir construire une matrice de distance. Elle sert également d'heuristique afin de minimiser les erreurs de comparaisons entre les images reçues en entrées. La structure de nos mesures, par exemple, les poussent à reconnaître 3 comme étant un 8 si les images sont comparées en partant du 3. Le cas contraire n'engendre, par contre, aucun problème de ce type. Ceci est dû au fait que la forme d'un 3 est comprise dans la forme d'un 8, mais pas l'inverse.
- Localiser le centroïde des deux listes de pixels foncés et les recentrer au même point, soit l'emplacement du centroïde de la liste1. Cette étape sert à aligner le plus possible les agrégats de pixels foncés qui constituent les deux images comparées (c.-à-d. centrer les deux chiffres l'un sur l'autre).

Le reste du code est spécifique à chaque mesure et leur description respective est présentée ci-bas.

Buffered pixel mapping (BPM)

Suite à la base commune de code, le code spécifique au BPM se charge de:

- Ajouter une couche de pixels foncés qui entoure l'agrégat de pixels foncés déjà contenu dans la liste2 (la liste de pixels la plus courte). ''L'épaisseur'' de la couche ajoutée représente un second hyperparamètre nommé ''Buffer radius''. Cette étape sert à intégrer une ''marge d'erreur'' autour du deuxième chiffre comparé afin d'augmenter les probabilités que les pixels foncés des deux chiffres comparés se chevauchent (c.-à.d. qu'ils partagent les mêmes coordonnées sur leur images respectives).
- Transformer les coordonnées à deux dimensions des pixels dans les deux listes de pixels foncés en scalaires à une dimension tout en préservant toute l'information contenue dans les coordonnées. Cette étape ''aplati'' les listes, qui sont sous forme de matrice, en vecteurs afin de faciliter l'étape suivante.
- Calculer le pourcentage de pixels foncés dans la liste1 qui ne sont pas également dans la liste2 (c.-à-d. vérifier pour chaque pixel dans la liste 1 si un pixel foncé existe au même emplacement dans la liste2). Le résultat engendré par ce calcul représente le pourcentage de pixels de la liste1 qui ne chevauchent pas les pixels de la liste2. Deux images similaires, dont la plupart des pixels se chevauchent, vont donc engendrer un petit résultat, alors que deux images très différentes vont engendrer un grand résultat.

Average Nearest Pixel (ANP)

Suite à la base commune de code, le code spécifique au ANP se charge de:

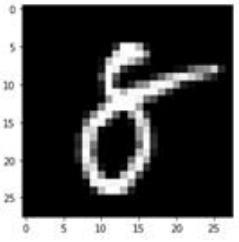
- Pour chaque pixel foncé dans la liste1, calculer la distance au pixel de la liste2 avec les coordonnées les plus rapprochées. La distance calculée représente la distance euclidienne au carré appliquée aux coordonnées des paires de pixels dans le cadre de l'image de taille 28x28 pixels. Étant donné

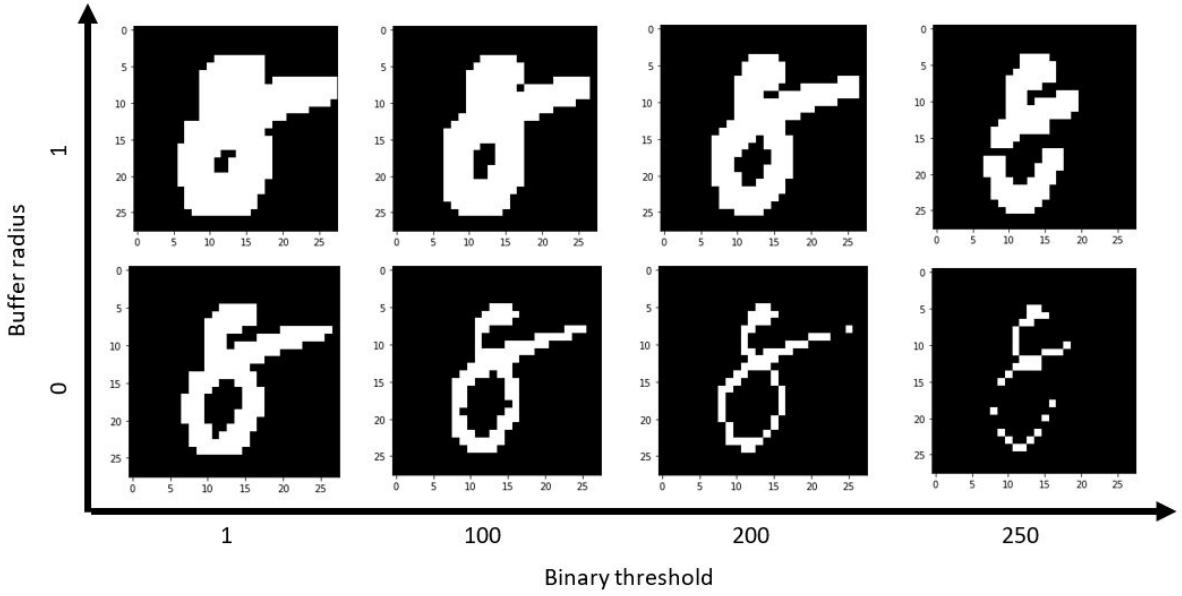
que cette distance est au carré, elle pénalise fortement les pixels situés loins l'un de l'autre. Une fois, la distance minimale pour chaque pixel foncé de la liste1 aux pixels de la liste2 calculée, la moyenne de ces distances minimales sur la liste1 est renvoyée comme résultat. Deux images similaires, dont les pixels sont généralement rapprochés, vont donc engendrer un petit résultat, alors que deux images très différentes vont engendrer un grand résultat.

Hyperparamètres

Nos deux mesures de similarité partagent un hyperparamètre commun, le ''binary threshold'' qui détermine la sélection de pixels foncés selon une valeur seuil reçue en entrée. La mesure BPM comporte également un second hyperparamètre, le ''buffer radius'' qui ajoute une couche de pixels foncés autour des pixels foncés existants. ''L'épaisseur'' de la couche dépend d'une valeur reçue en entrée. L'effet de la variation de ces deux hyperparamètres est illustrée ci-bas afin d'aider à une compréhension intuitive de leur effet sur une image prise en entrée.

Image originale:





Hypothèses de performance
Afin d'obtenir des performances supérieures à celle de la distance euclidienne, nous avons eu recours à deux stratagèmes majeurs:

Minimisation des erreurs causées par un décalage entre deux images comparées:
La sélection exclusive des pixels foncés (représentant une valeur de gradient de gris plus grande que le seuil défini avec hyperparamètre ''binary threshold'') jumelée avec l'alignement du centroïde des deux images (chiffres) considérées permet d'assurer un meilleur chevauchement des chiffres décalés. Ceci devrait se traduire en mesures de similarité plus exactes et aptes à minimiser les erreurs causées par des images décalées.

Minimisation des erreurs causées par un désalignement angulaire entre deux images comparées:
Nous avons remarqué que plusieurs images MNIST présentent des chiffres inclinés (c.-à.d. requérant une rotation afin de paraître droits), ce que nous jugeons nuire à la qualité des estimations de similarité.

Dans le cas de la mesure BPM, nous avons tenté de remédier à ce problème en introduisant la possibilité de créer une couche supplémentaire de pixels foncés autour des pixels sélectionnés avec l'hyperparamètre ''binary threshold''. L'épaisseur de cette couche introduit un second hyperparamètre ''buffer radius''. L'idée est d'introduire un ''error buffer'' autour d'un des deux chiffres comparés afin de contrer un possible désalignement angulaire entre ceux-ci.

Dans le cas de la mesure ANP, au lieu de comparer les valeurs des pixels situés au même emplacement sur les deux images considérées, nous calculons, pour chaque pixel foncé d'une image, la distance (euclidienne au carré) du pixel foncé dans la seconde image le plus près de son emplacement. Ce stratagème permet de passer d'une technique de type ''hit or miss'', soit la comparaison pixel à pixel, à une technique plus flexible qui pénalise moins les chiffres ayant un désalignement angulaire.

Résultats

Cette section présente les résultats obtenus en comparant la performance rattachée à l'utilisation de différentes mesures sur la gamme d'algorithmes considérés.

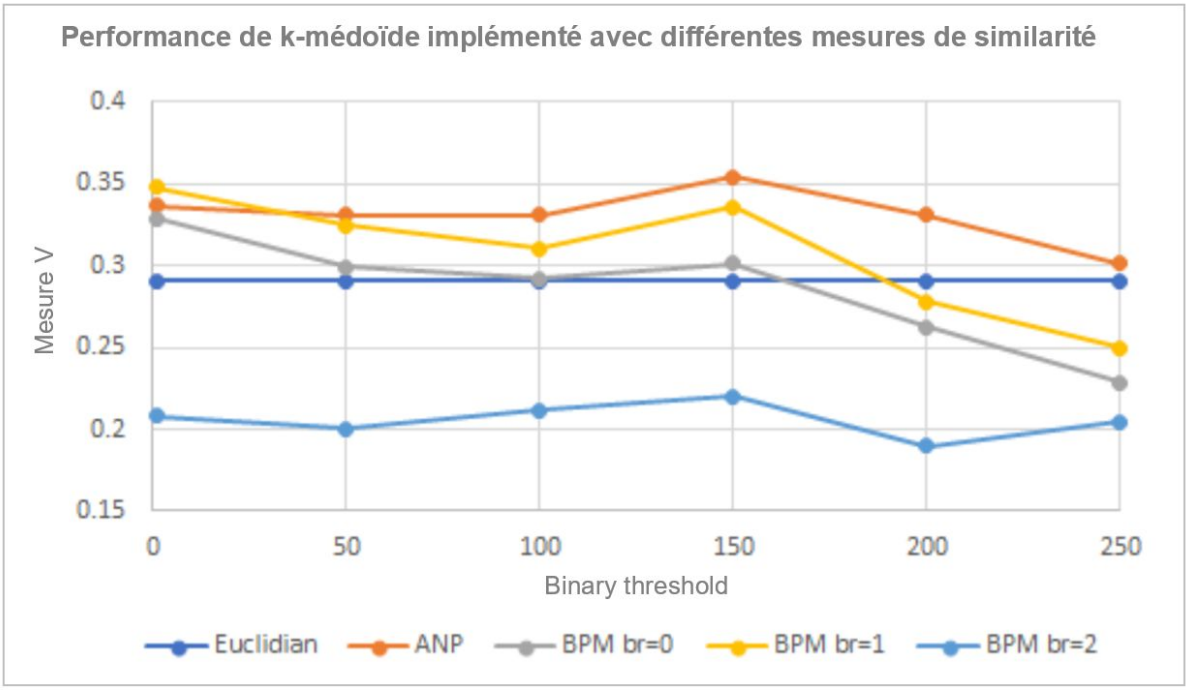
Légende

- ANP: Average Nearest Pixel
- BPM: Buffered pixel mapping
- br: buffer radius

k -médoïde

Mesure V pour k-médoïde

binary-threshold	Euclidean	ANP	BPM br=0	BPM br=1	BPM br=2
1	0.290827663	0.336744593	0.329027157	0.347531246	0.20780522
50	0.290827663	0.331228299	0.299464044	0.324867964	0.200220001
100	0.290827663	0.331228299	0.292348654	0.310633342	0.21169268
150	0.290827663	0.354430704	0.301115167	0.335734859	0.220163199
200	0.290827663	0.331228299	0.262423498	0.278178171	0.189601958
250	0.290827663	0.301510016	0.228634411	0.249681095	0.204554359

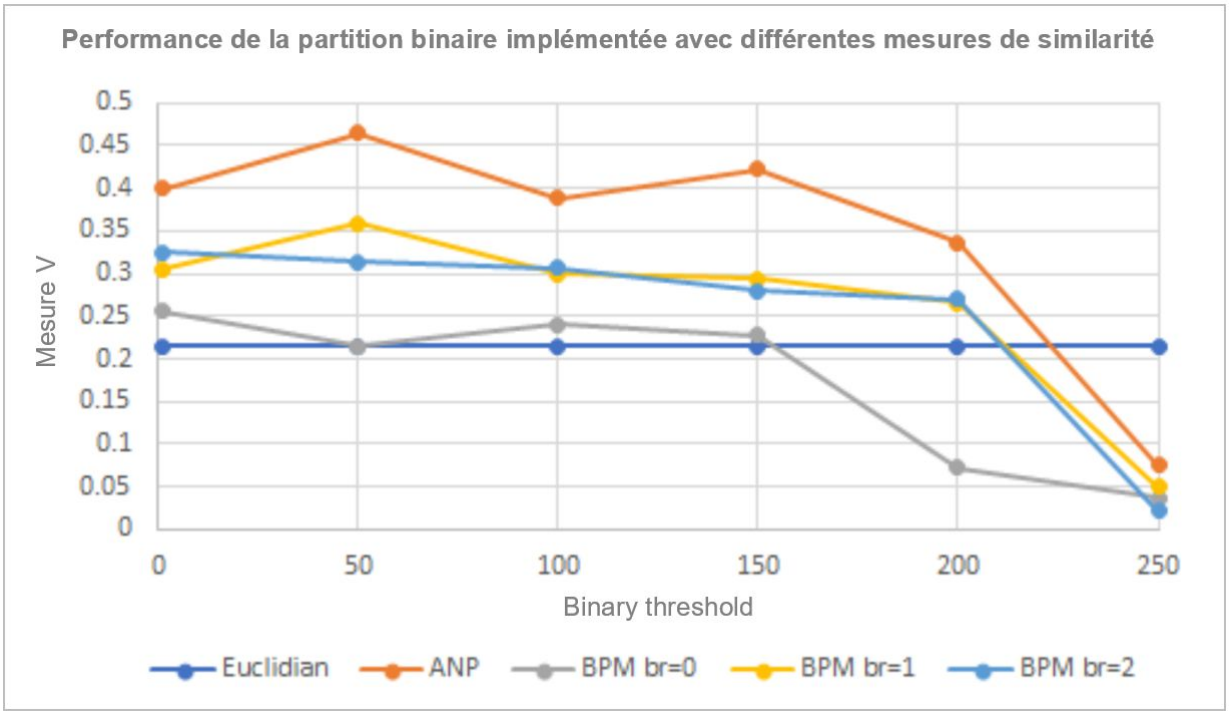


Dans le cas de l'algorithme k-médoïde, l'utilisation des deux mesures engendre une meilleure performance que la distance euclidienne pour la plupart des combinaisons d'hyperparamètres testées. La meilleure performance, soit une mesure V de 0.3544, est obtenue à l'aide de la mesure ANP avec un binary threshold de 150.

Partition binaire

Mesure V pour partition binaire

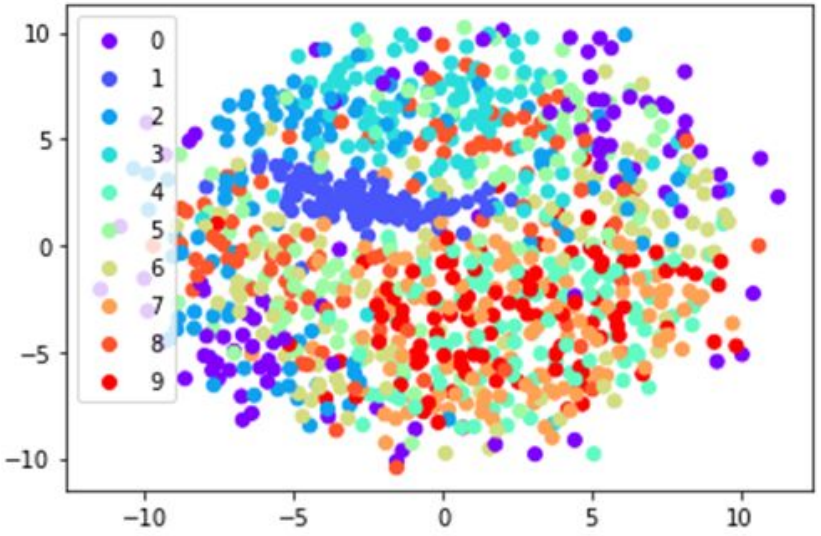
binary-threshold	Euclidean	ANP	BPM br=0	BPM br=1	BPM br=2
1	0.215498795	0.400297576	0.255382	0.305455751	0.325643792
50	0.215498795	0.464600359	0.215448345	0.35880717	0.313849874
100	0.215498795	0.388099745	0.239696426	0.299872888	0.306325859
150	0.215498795	0.421671368	0.228000498	0.29431131	0.279161691
200	0.215498795	0.336904082	0.07167339	0.265331517	0.270250822
250	0.215498795	0.075545109	0.036971248	0.04957951	0.021373318



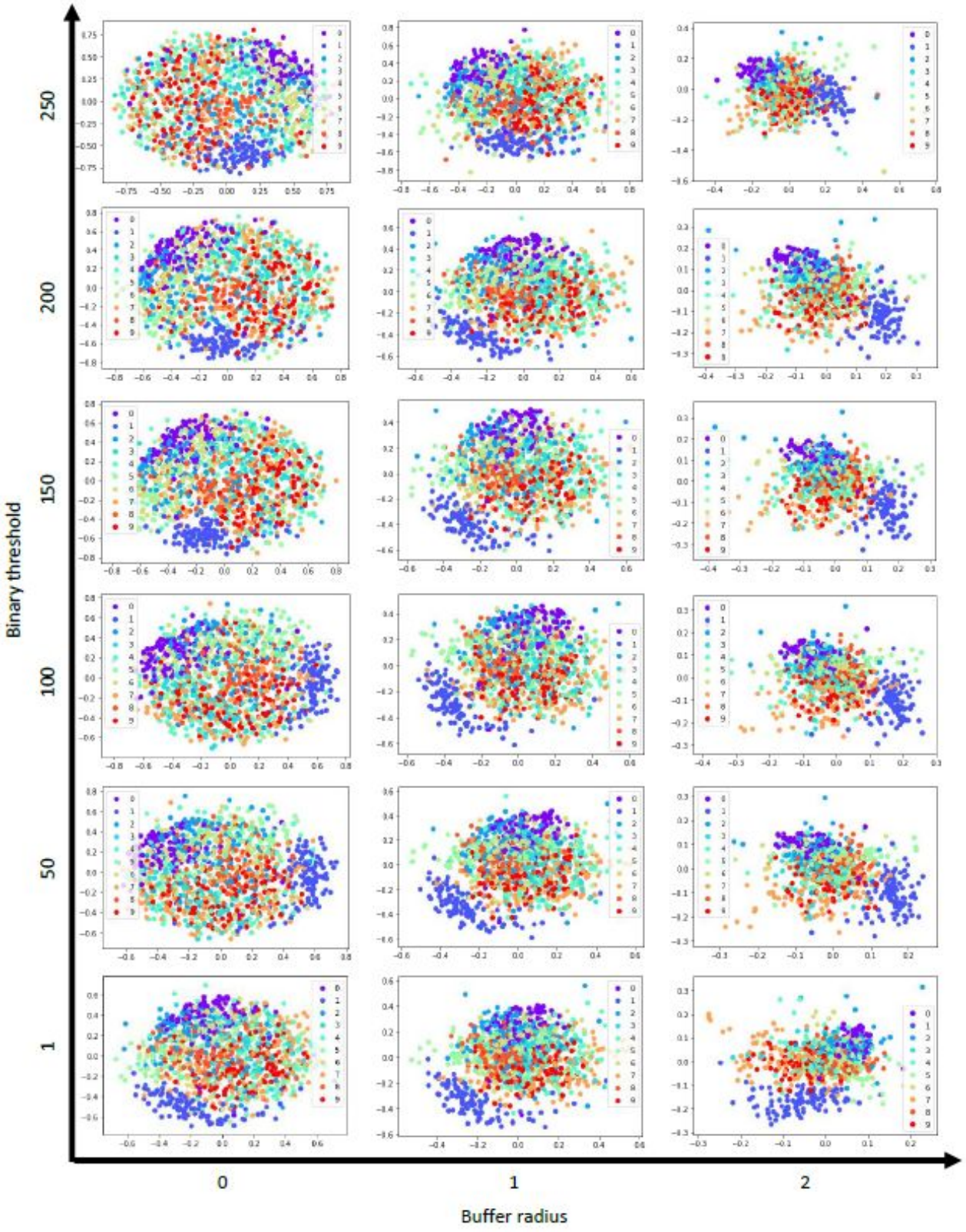
Dans le cas de l'algorithme de partition binaire, l'utilisation des deux mesures engendre une meilleure performance que la distance euclidienne pour la plupart des combinaisons d'hyperparamètres testées. La meilleure performance, soit une mesure V de 0.4646, est obtenue à l'aide de la mesure ANP avec un binary threshold de 50.

PCoA

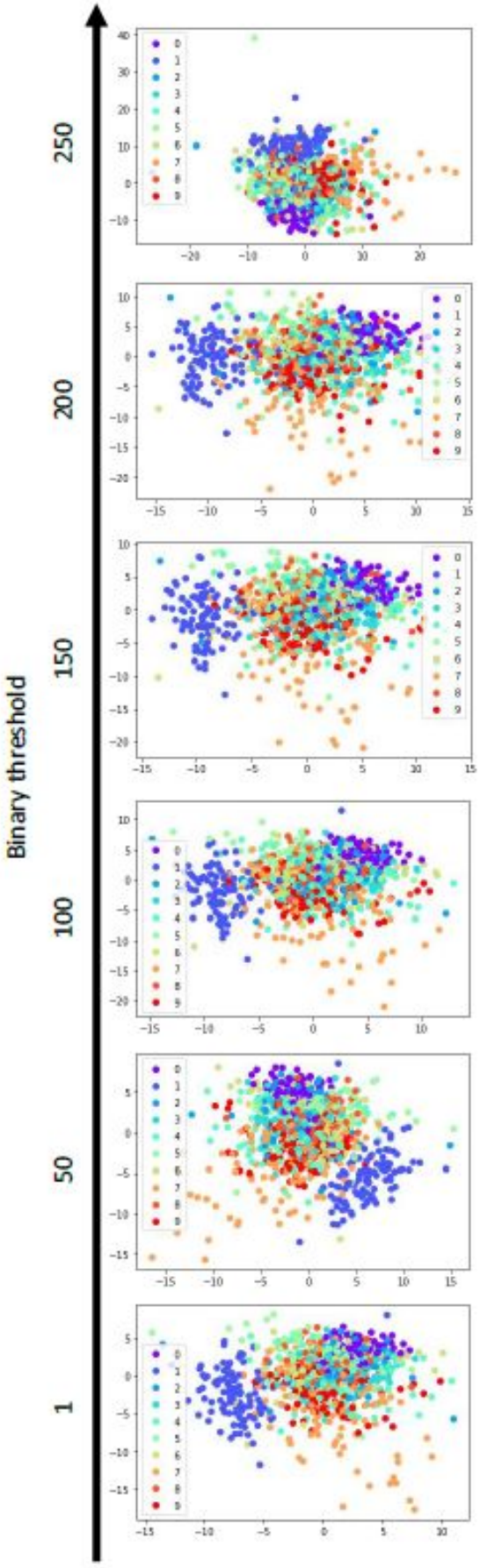
PCoA avec 2 composantes implémenté avec la distance euclidienne



PCoA avec 2 composantes implémenté avec BPM



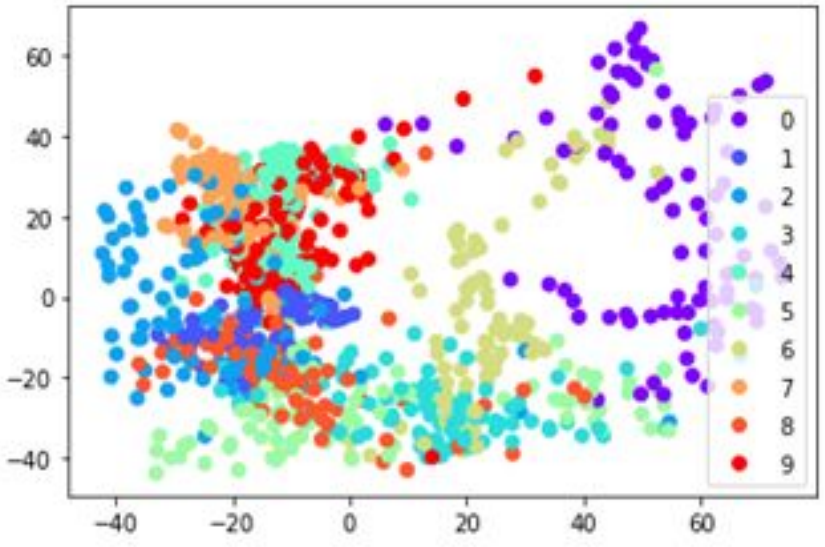
PCoA avec 2 composantes implémenté avec ANP



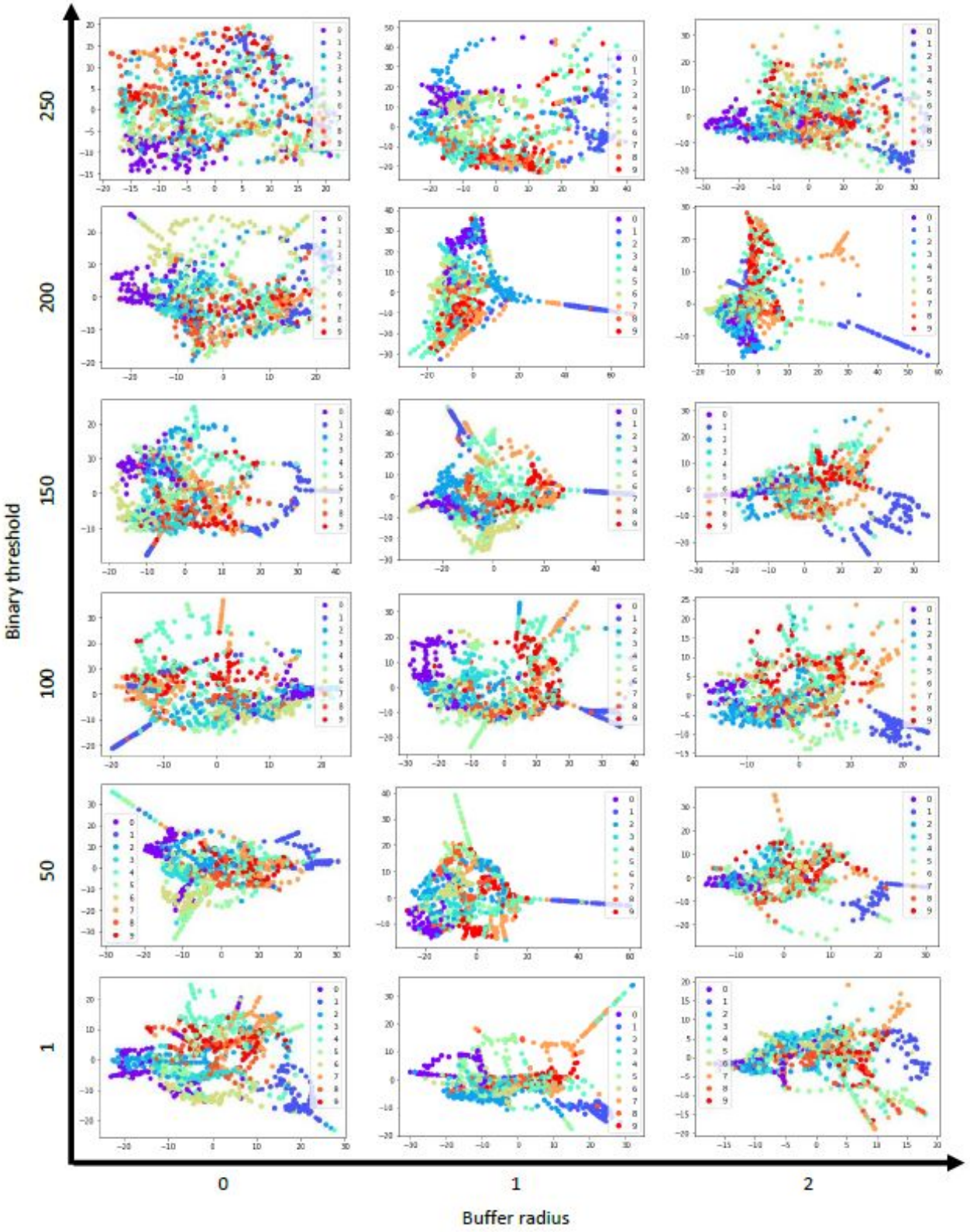
Dans le cas de l'algorithme PCoA, l'évaluation de la performance repose sur des observations visuelles qualitatives qui sont moins fiables et rigoureuses que des résultats quantitatifs. L'utilisation des deux mesures semble engendrer une meilleure performance que la distance euclidienne pour la plupart des combinaisons d'hyperparamètres testées compte tenu que des agglomérations distinctes semblent être formées pour les exemples étiquetés 0 et 1. l'utilisation de la distance euclidienne semble seulement former une agglomération distincte pour les exemples étiquetés 1. La meilleure performance semble être obtenue à l'aide de la mesure ANP avec un binary threshold de 50 ou 100 compte tenu qu'il est, dans une certaine mesure, possible de visuellement dissocier les agglomérations formées pour les exemples étiquetés 0, 1, 3, 5 et 7.

Isomap

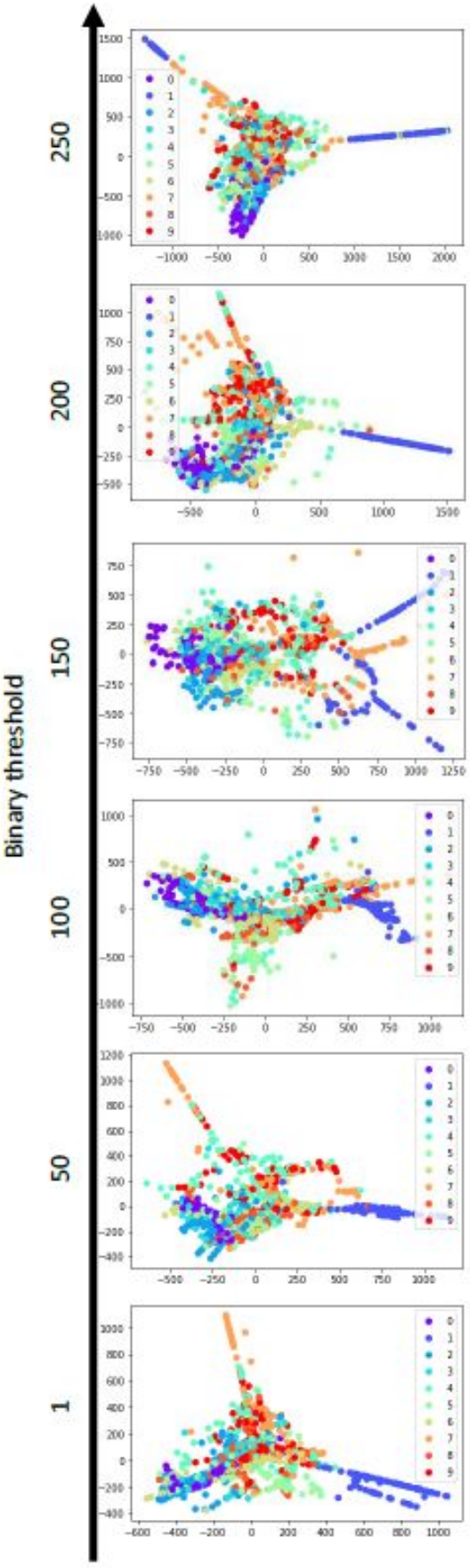
Isomap avec 2 composantes implémenté avec la distance euclidienne



Isomap avec 2 composantes implémenté avec BPM



Isomap avec 2 composantes implémenté avec ANP

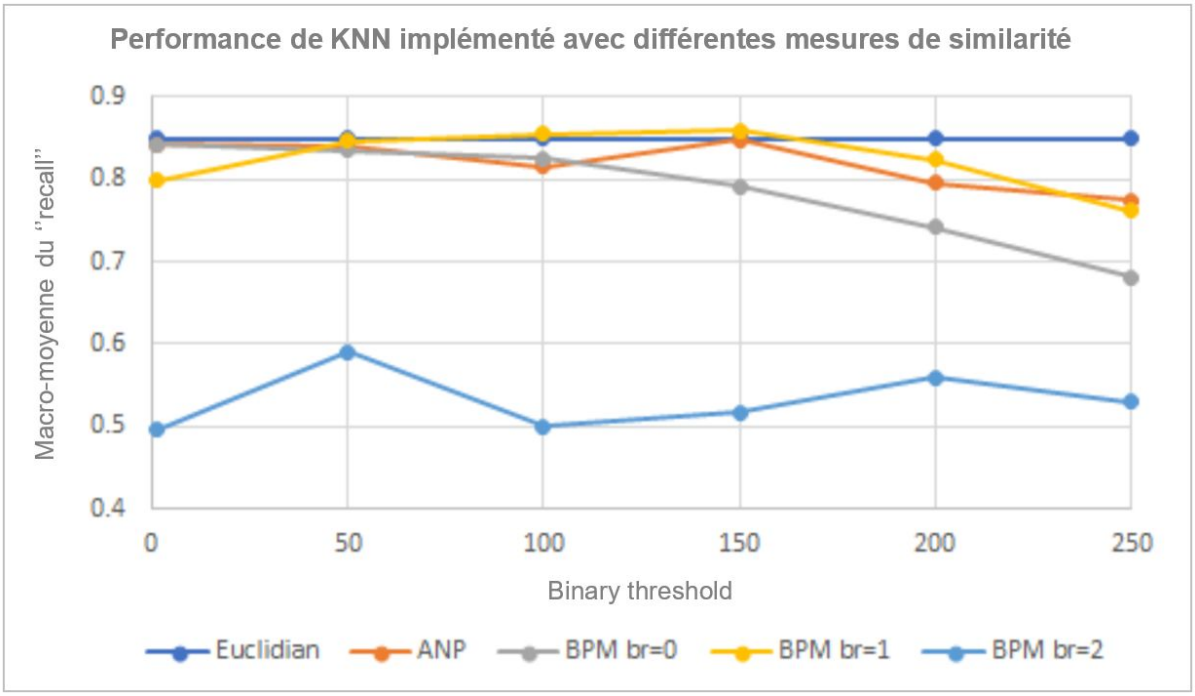


Dans le cas de l'algorithme Isomap, l'évaluation de la performance repose sur des observations visuelles qualitatives qui sont moins fiables et rigoureuses que des résultats quantitatifs. L'utilisation de la distance euclidienne semble engendrer la meilleure performance compte tenu qu'elle produit des agglomérations distinctes pour les exemples de chaque classe, bien que que ces agglomérations se chevauchent dans un même espace. L'utilisation des mesures BPM et ANP semblent parfois produire des agglomérations distinctes uniquement pour certaines classes spécifiques, mais jamais pour l'entièreté des classes.

KNN

Macro-moyenne du ''recall'' pour KNN

binary-threshold	Euclidean	ANP	BPM br=0	BPM br=1	BPM br=2
1	0.849457085	0.842167414	0.841606211	0.797674565	0.496450919
50	0.849457085	0.839650117	0.83484761	0.84553247	0.591239316
100	0.849457085	0.815024175	0.825591007	0.854720735	0.500876068
150	0.849457085	0.847378973	0.79154552	0.858614519	0.517232906
200	0.849457085	0.795303235	0.741670971	0.823208542	0.559711538
250	0.849457085	0.774356123	0.680566355	0.761496038	0.530235043



Dans le cas de l'algorithme KNN, l'utilisation des deux mesures engendre une meilleure performance que la distance euclidienne pour seulement une minorité des combinaisons d'hyperparamètres testées. La meilleure performance, soit une mesure macro-moyenne du score recall de 0.8586, est obtenue à l'aide de la mesure BPM avec un binary threshold de 150 et un buffer radius de 1. Il est important de noter que la différence entre la meilleure performance obtenue à l'aide de la mesure BPM, celle obtenue à l'aide de la mesure ANP, soit mesure macro-moyenne du score recall de 0.8474, et celle obtenue avec la distance euclidienne est très petite, voir même négligeable. Il n'est donc pas possible d'affirmer avec certitude que l'utilisation d'une de ces mesures a clairement engendré une performance meilleure que les autres.

Conclusion

Selon nos observations, les mesures qui ont engendré les meilleures performances pour chaque algorithme considéré étaient:

- K-médoïde: ANP avec un binary threshold de 150
- Partition binaire: ANP avec un binary threshold de 50
- PCoA: ANP avec un binary threshold de 50 ou 100
- Isomap: distance euclidienne
- KNN: incertitude entre la distance euclidienne, BPM avec un binary threshold de 150 et un buffer radius de 1, et ANP avec un threshold de 150

La mesure ANP a engendré les meilleures performances pour, au moins, 3 des 5 algorithmes considérés et semble être, en général, la mesure utilisée la plus performante.

Nous avons également observé une capacité générale de nos mesures à engendrer des meilleures performances que la distance euclidienne sur l'ensemble des résultats quantitatifs observés (c.-à-d. pour les algorithmes k-médoïde, partition binaire et KNN).

Discussion

Tel que mentionné dans la section méthodologie, un faille importante rattachée à l'usage d'un échantillon de 1 000 exemples tirés aléatoirement sans répétition sur le jeu de données complet MNIST est l'augmentation de la variance des modèles considérés (erreur d'estimation), soit la variabilité des résultats en fonction du jeu d'apprentissage fourni, due à l'utilisation d'un échantillon très petit. Afin d'obtenir des résultats de test plus fiables, une bonne pratique serait de répéter les tests sur chaque modèles plusieurs fois pour toutes les mesures utilisées et de prendre la moyenne des résultats afin de comparer les performances des différentes mesures.

Pour chaque algorithme considéré, nous avons évalué la performance sur 6 valeurs de l'hyperparamètre binary threshold (jumelés à 3 valeurs de l'hyperparamètre buffer radius dans le cas de la mesure BPM) pour chaque mesure. Bien que nous jugeons le spectre des valeurs utilisées adéquat, étendre les tests à davantage de valeurs pour le binary threshold pourrait potentiellement révéler des résultats plus performants pour nos deux mesures.

Faute de temps, nous voulions implémenter une troisième mesure de similarité originale pour laquelle nous aurions fait une translation du centroïde d'une image (chiffre) vers celui de la deuxième image (comme dans nos deux mesures implémentées) suivie d'une mesure de distance euclidienne de pixel à pixel. Nous avions également envisagé de développer un moyen de faire la rotation d'une des deux images afin de les aligner le plus possible, ce qui aurait probablement amélioré les performances engendrés par les mesures.