UNIVERSITÉ DE GENÈVE

ANALYSE ET TRAITEMENT DE L'INFORMATION
14X026

TP 3: k-NN, PCA, FCA

Author: Julien Python

E-mail: julien.python@etu.unige.ch

October 25, 2020

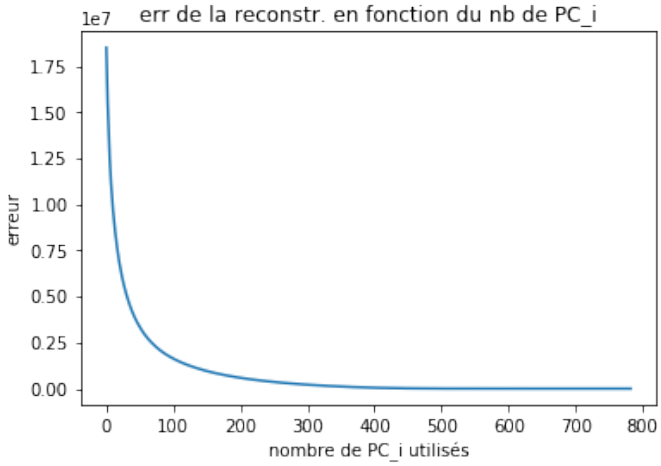
UNIVERSITÉ
DE GENÈVE
FACULTÉ DES SCIENCES
Département d'informatique

Remarque: J'ai d'abord effectué les exercices de PCA avec des boucles, mais le temps de chargement était trop long et je n'arrivais pas à obtenir des plots pour répondre aux questions de la série. J'ai donc par la suite effectué ces calculs dans des arrays, mais je joins mes deux types de codes pour le rendu.

Exercice 1

Partie 1)

La figure ci-dessous représente le graphe de la fonction err(X,m) pour m allant de 1 à 784, où m est représenté sur l'axe x, par rapport à l'erreur sur l'axe y et X est un ensemble d'images. Il est important de noter que les PC_i sont triés par ordre décroissant. On remarque logiquement que plus on utilise de PC_i, plus notre erreur est petite, mais il est intéressant de noter que la différence de l'erreur est très faible par exemple si on prend les 600 premiers ou les 780 premiers. Donc selon l'erreur qu'on est prêt à accepter, on peut réduire la taille de nos données avec lequels nous travaillons en utilisant le graphe ci-dessous.



Pour les erreurs:

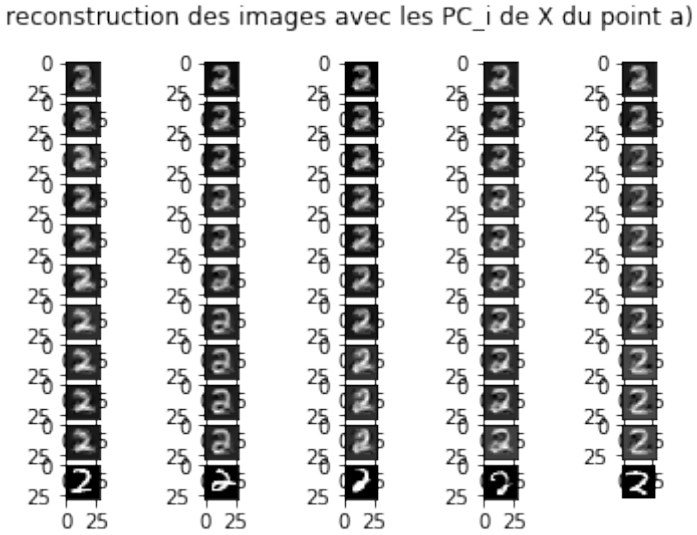
- pour avoir une erreur inférieure ou égale à 50%, il faut utiliser les 581 premiers PC_i

- pour avoir une erreur inférieure ou égale à 5%, il faut utiliser les 585 premiers PC_i

- cependant avoir une erreur de 0% n'est pas possible car même en utilisant tous les PC_i on a une erreur de: 5.5e-23 qui est dû aux approximations numérique (même si ce nombre est très proche de 0).

Partie 2)

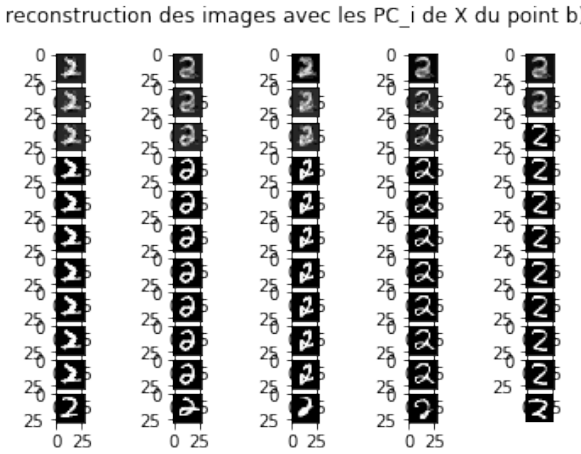
Sur la figure ci-dessous, chaque colonne correspond à une image, la colonne 1 correspond à m=1, la colonne 2 correspond à m=2, de même jusqu'à 10. La 11ème ligne représente quant à elle les images originelles de chacune des colonnes.



On observe sur ce plot que pour m=1 (la première ligne), chacunes des 5 images se ressemblent beaucoup, et on remarque qu'avec la croissance des m, les images se rapprochent de plus en plus de l'image originale. On peut noter que pour la 4ème colonne, les images ne se rapprochent pas vraiment du deux écrit. Ceci peut s'expliquer par le fait que ce 2 original ne ressemble pas beaucoup à un vrai deux (ou à la plupart des 2 de notre base de données), et donc sa reconstruction est plus problématique.

Ainsi, plus le nombre de principal components augmente, plus on se rapproche de l'image origniale. On note tout de même sur la 5ème colonne un certain flou sur les pixels noirs de l'image qui ressemblent à du noise. Il pourrait être réduit pour un résultat plus clair. On a donc que si k (nomvre de PC) est petit, l'image reconstruite ne ressemble pas à l'image originale, mais en augmentant ce dernier elle y ressemble de plus en plus.

Remarque: si on prends comme principal components ceux de la partie 2 de l'exercice, les résultats sont bien meilleurs. De plus, j'ai calculé les distances entre les images images reconstruites et originales pour ces deux figures dans le notebook python, et la différence est flagrante.



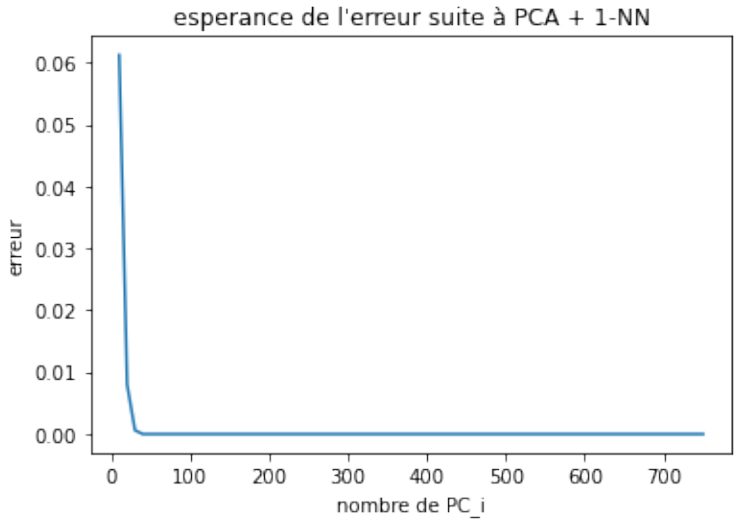
Exercice 2

Partie 1)

Nombre de tests réussis sur 5000: 4918 , ce qui donne un pourcentage de: 0.9836. C'est un résultat qui m'a surpris car en prenant seulement le premier voisin le plus proche on a déjà un taux assez élevé de réussite.

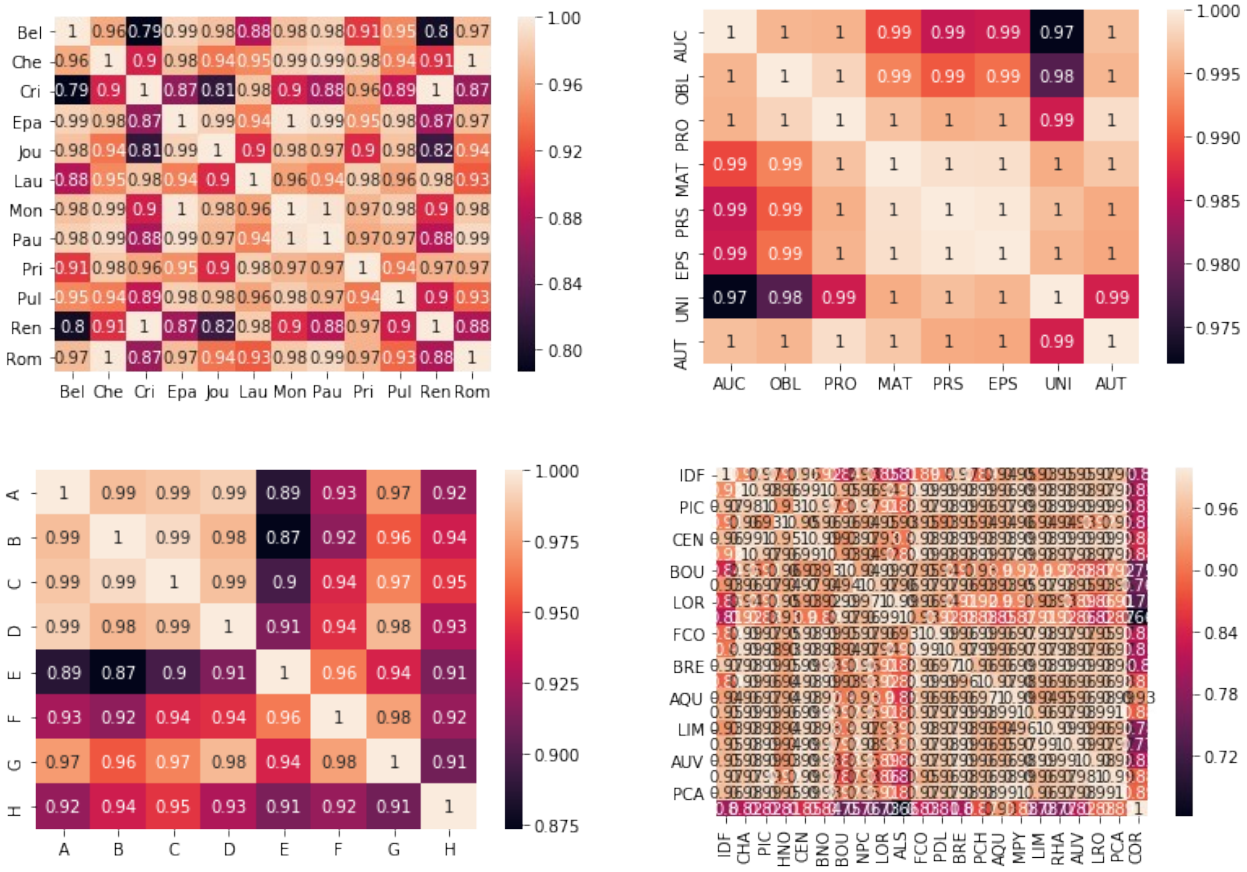
Partie 2)

On remarque sur la figure ci-dessous que l'erreur est relativement grande pour un nombre de PC_i faible (< ~40), mais dès qu'on a passé cette première partie où la pente est très raide, l'erreur est très faible, très proche de 0. Il est donc peu utilise de choisir les 784 PC_i pour ce problème alors qu'en en choisissant seulement 50 par exemple, on obtient le même résultat (ou un résultat très proche), tout en effectuant moins de calculs. Donc l'analyse effectuée dans cet exercice nous permet d'affirmer qu'en moyenne, prendre les 50 premiers PC_i est le choix optimal qui permet de réduire la taille des données avec lesquelles nous travaillons tout en ayant une erreur proche de zéro.



Exercice 3

Voici les quatres matrice de correlation, respectivement pour df1 (colonnes et lignes), pour df2 (colonnes et lignes).



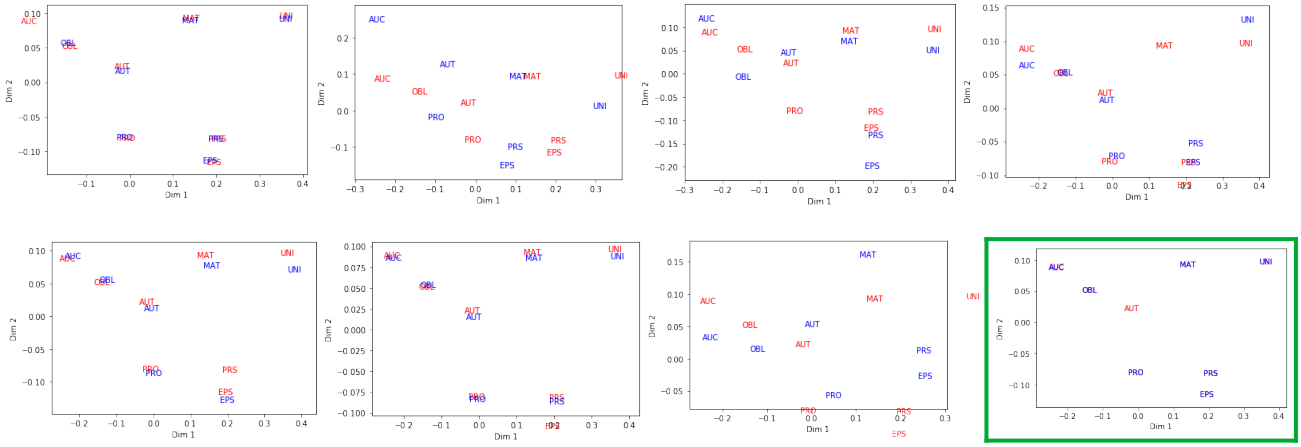
Rappel sur le coefficient de correlation: si le coefficient est proche de 1, cela signifie que les données sont très corrélées, par exemple si cor(A,B)=~ 1, si A augmente alors B a de forte chance d'augmenter, de même avec la diminutier. Par contre, si cor(A,B)=0, cela signifie que nos deux variables aléatoires A et B sont indépendantes, et donc si A augmente (ou diminue), cette information ne nous donne aucun indice sur le comportement de B. Je ne détaille pas si le coefficient de corrélation vaut -1, car ce n'est pas présent dans notre exercice, mais il s'agit de l'inverse d'une correlation de 1.

On remarque donc que ces quatres matrices sont assez corrélées (entre 0.75 et 1), avec toutefois des variations clairement visibles grâce au package seaborn. Ceci nous semble logique car il s'agit de données similaires mais sur différentes régions, ce qui peut expliquer ces différences.

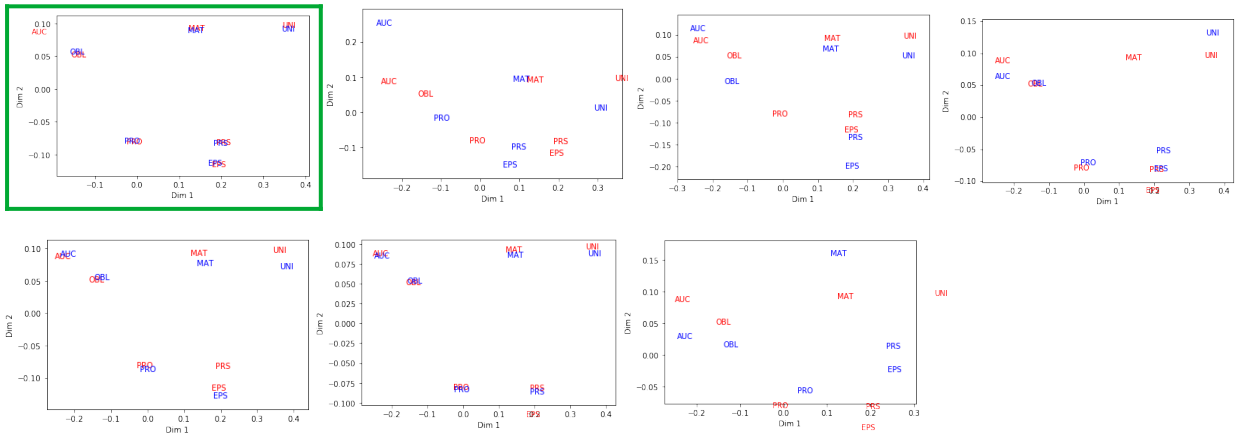
Ainsi, si on souhaite réduire la taille des données en perdant le moins d'informations possible, on va cherchercher à supprimer la variable aléatoire qui a le coefficient de corrélation le plus élevé avec ses voisins afin que lorsqu'on la supprime, on perde le moins d'informations possible. C'est ce qu'on fait dans la suite de cet exercice.

On peut faire la remarque que si on avait eu une 9ème variable aléatoire indépendante des huit autres, si on l'avait supprimée, on aurait perdu beaucoup d'informations.

Ainsi, comme on l'observe sur les graphes ci-dessous, on enlève AUT en premier, choix confirmé car AUT a un coefficient de correlation proche de 1 avec toutes les autres variables alétoires.



Ensuite, on enlève AUC: (même explication que ci-dessus)



Et enfin, on enlève EPS : (même explication que ci-dessus)