Machine Learning en Python

Martelli Gino, Senis Tahitoa, Vieville Sébastien

Introduction

Le Machine Learning permet à un système d'apprendre à partir des données afin d'effectuer des prédictions sur des nouvelles données.

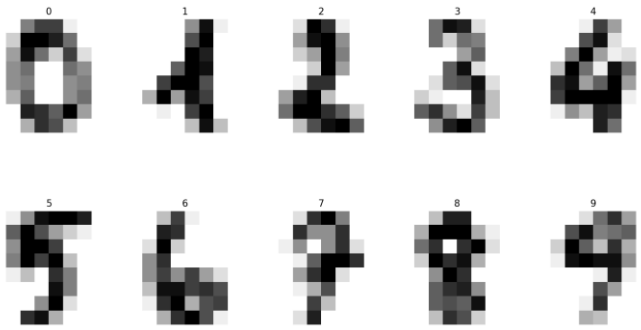
Objectif : La machine doit pouvoir déterminer quel chiffre est sur une image de 8*8 pixels qu'elle n'a jamais vus.

Méthode : On va utiliser l'apprentissage supervisé : on fournit à la machine des exemples avec leur réponses et elle va apprendre à faire des prédictions correctes. Chaque image représente un chiffre manuscrit de 0 à 9 et est associée à une étiquette (le chiffre correct). Le modèle apprend à reconnaître les motifs visuels et à prédire la bonne classe. Il faut donc trouver une fonction qui prend en entrée une image d'un chiffre manuscrit et qui renvoie en sortie le chiffre correspondant.

Étapes :

- Préparation des données
- Features engineering
- Division des données
- Apprentissage et validation

Préparation des données



On va utiliser un dataset de sklearn (librairie en Python) qui contient 1797 images en 8*8 pixels de chiffres manuscrits, où chaque pixel à une valeur entre 0 et 16 qui correspond à une intensité. Une image est donc une matrice avec 0 pour noir et 16 pour blanc. Voici une image aléatoire prise dans chaque classe. Voici la distribution de notre base de données

On va d'abord normaliser les données pour éviter les différences de dynamique dans les valeurs. On utilise MinMaxScaler() pour avoir des données entre 0 et 1.

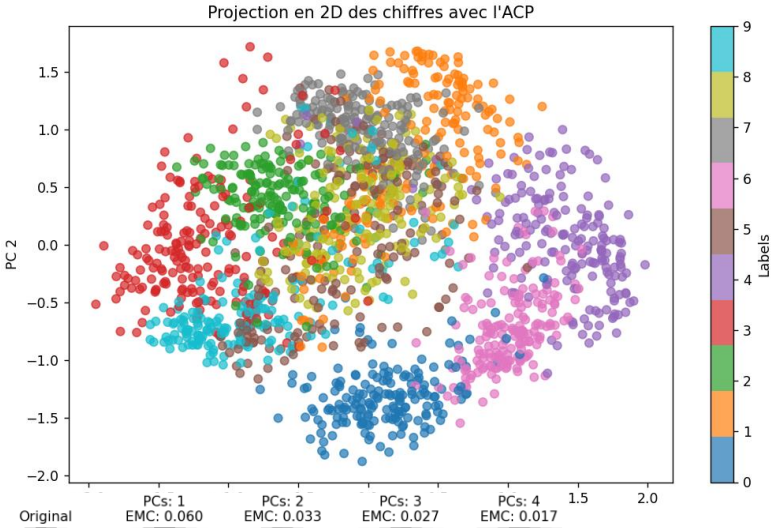
Features engineering

Les images ont 64 pixels mais elles n'ont pas toutes des informations utiles et exploitables. On va essayer de conserver seulement des valeurs pertinentes afin de simplifier le modèle pour l'apprentissage, en passant d'images en 64 dimensions à des images en 24 dimensions. Ainsi, le modèle identifiera des modèles simples. De plus, une image brute est trop complexe pour être directement analysée. On prend 20 valeurs en utilisant l'analyse en composantes principales (ACP), 3 valeurs qui correspondent à 3 zones de l'image et une dernière valeur pour la détection des contours.

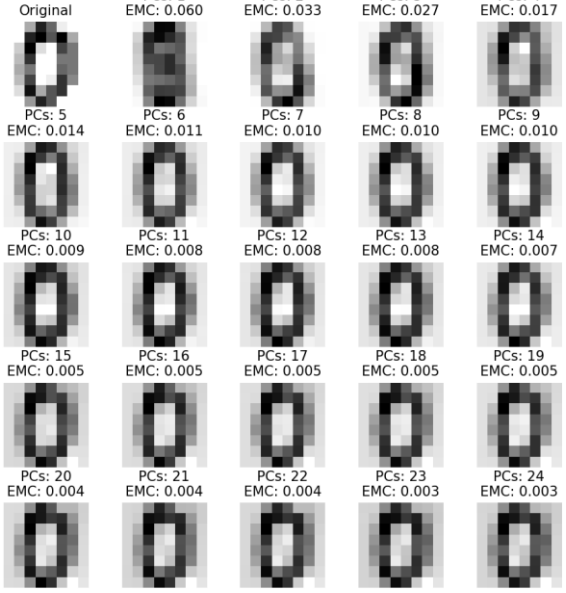
Analyse en composantes principales :

L'analyse en composantes principales (ACP) est une méthode de réduction de dimension qui transforme des variables initiales corrélées en un nouveau jeu de variables non corrélées (c'est à

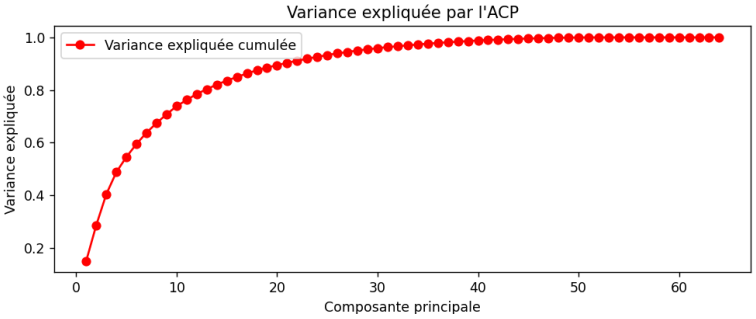
dire orthogonales les unes par rapport aux autres). Elle est utilisée dans ce projet pour réduire la dimension des images tout en conservant le plus d'informations.



En 2 dimensions, on peut déjà voir des regroupements de chiffres qui correspondent à la même classe, notamment les 0. Il y a beaucoup de classes différentes dans la partie supérieure du graphique : utiliser seulement 2 composantes principales n'est pas suffisant pour prédire correctement le label d'une image. On peut visualiser (avec comme exemple 0) que plus on prend de composantes, plus l'erreur diminue (erreur moyenne au carré).



Le choix du nombre de composantes principales est offert à nous. On peut choisir en s'aidant du graphe de la variance cumulé expliquée :



On choisit de prendre 20 composantes principales afin d'expliquer entre 90 et 95% des variances et d'avoir une erreur de 0,04.

Partitionnement en zones : On va diviser l'image en plusieurs régions pour mieux analyser la structure : région du haut (lignes 1,2 et 3), du milieu (lignes 4 et 5) et du bas (lignes 6,7 et 8). On prend la moyenne d'intensité de chacune des régions.

Détection des contours : On va appliquer un filtre de Sobel afin d'identifier les contours et de mettre en évidence les structures principales de chaque chiffre. Avec ce filtre on va trouver deux variables : la première mesure combien varie l'intensité horizontalement et la deuxième verticalement. En faisant la moyenne entre ces deux variables nous allons obtenir la dernière composante que nous utiliserons pour les modèles de machine learning.

Concatenation des features : On crée finalement une matrice finale qui contient les 3 features. On passe donc d'une base de données de taille 1797*64 à 1797*24. Ces features sont utiles car elles réduisent le bruit, la redondance des données et le nombre de caractéristiques, tout en améliorant la robustesse de la classification.

Division des données

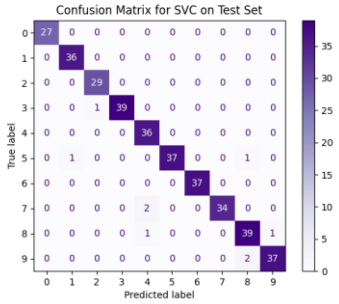
Dans notre cas, diviser les données est utile pour tester le modèle avec des données inédites pour le modèle. Il y a plusieurs façons de le faire : entraînement/test, cross-validation, stratification. Cette étape est importante car une mauvaise répartition peut fausser les résultats. Un ensemble

trop petit sur l'ensemble des tests limite la fiabilité des résultats, et un ensemble trop petit pour l'entraînement limite la précision du modèle.

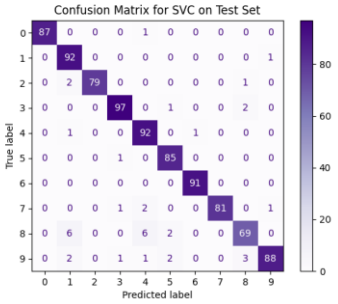
Pipelines

Les pipelines sont des structures qui permettent de chainer plusieurs étapes de traitement de donnée (scaling, type de modèle, handcrafted features) dans un flux unique ce qui permet d'éviter les erreurs et garantit une bonne reproductibilité.

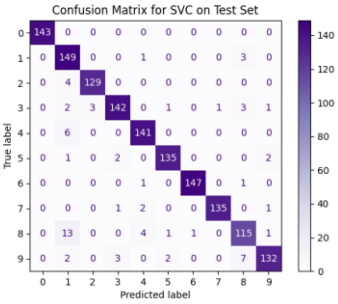
Confusion Matrix



80/20 % de train/test



50/50 % de train/test

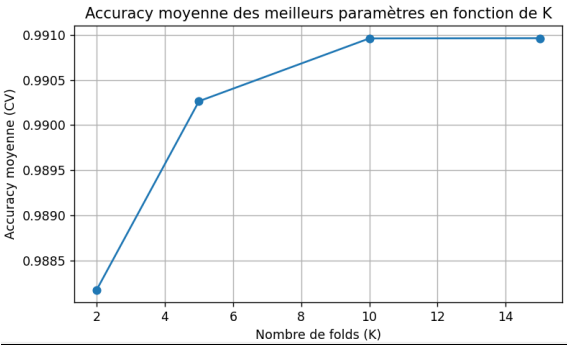


20/80 % de train/test

On remarque assez facilement que plus l'échantillon de training est grand, moins il y aura d'erreur sur les tests qu'on fera, c'est-à-dire plus précise sera notre estimation des chiffres. Notamment pour la dernière matrice de confusion où il y a beaucoup de prédiction de 1 alors que ce n'est pas le vrai label.

SVC model :

On arrive donc au premier modèle de machine learning, le SVC (support vector classification). Le SVC consiste à tracer le meilleur hyperplan possible pour séparer toutes les classes. Nous utilisons la fonction "GridSearchCV" pour trouver les paramètres optimaux (nombre de composantes PCA, type de kernel, valeur de C, méthode de scaling) pour avoir la meilleure précision. Avec les paramètres optimaux nous obtenons une précision de 99,3% sur l'échantillon de test.



Pour obtenir ce graphique, nous avons fait une boucle calculant la précision de la méthode. On peut voir que plus la valeur de K est grande plus la précision sera élevée.

OV- CLASSIFIER :

Critère	OvO (One vs One)	OvR (One vs Rest)
Score de test	0.969	0.965
Nombre de classifieurs	45	10
Temps d'entraînement (s)	0.762	0.529
Impact	Plus précis pour des petits datasets mais prend plus de temps car il fait beaucoup de classes.	Meilleur en général pour les grands datasets, il prend également moins de temps à se faire car il calcul moins de classe, cependant on perd en précision.

One-vs-One (OvO), qui entraîne un classifieur par paire de classes, et One-vs-Rest (OvR), qui oppose chaque classe à toutes les autres.

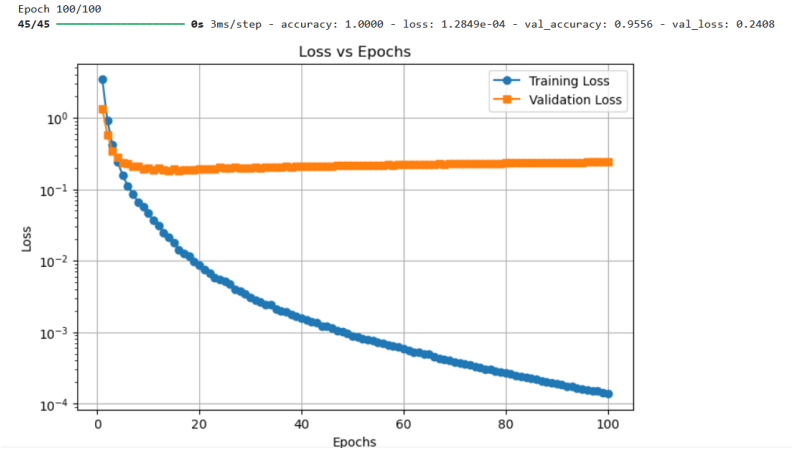
Neural Network :

Pour le réseau de neurones nous avons utilisé le modèle “sequential” qui est un modèle linéairement empilé, c’est à dire que chaque couche de neurones est uniquement connectée à sa couche précédente et à sa couche suivante il n’y a pas d’autre connexions entre les couches comme il pourrait en avoir avec d’autre modèle. Pour la couche d’entrée et la couche caché on utilise la fonction d’activation “relu” qui renvoie 0 si l’entrée est négative et ne change pas la valeur si l’entrée est positive.

Pour la couche de sortie on va utiliser la fonction d’activation “softmax” qui transforme la sortie brute en une probabilité dont la somme vaut 1.

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

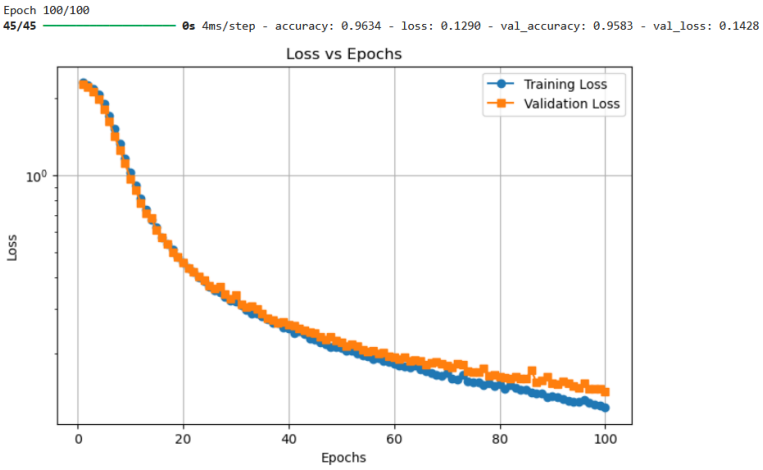
Dans un premier temps nous avons appliqué ce modèle sur les données brut c’est à dire les 64 pixels de chaque image. Après avoir compilé et entrainé le modèle sur 100 epochs nous obtenons une précision de 95,56% avec une perte de 24,08% sur l’échantillon (X_test , y_test).



On peut voir ici que la perte pour l’échantillon d’entrainement et de validation baisse rapidement au début, cohérent car le modèle apprend. Puis les pertes commencent à stagner vers la dixième epoch pour le (X_test , y_test) alors que l’échantillon d’entrainement continue de s’améliorer. Ce qui se voit par un grand écart entre les deux courbes à partir de la 15ème epoch, cela montre un problème d’overfitting (le modèle continue de devenir meilleur pour prédire les images qu’il connait mais ne s’améliore pas sur ce qu’il ne connait pas).

On peut voir ici que la perte pour l’échantillon d’entrainement et de validation baisse rapidement au début, cohérent car le modèle apprend. Puis les pertes commencent à stagner vers la dixième epoch pour le (X_test , y_test) alors que l’échantillon d’entrainement continue de s’améliorer. Ce qui se voit par un grand écart entre les deux courbes à partir de la 15ème epoch, cela montre un problème d’overfitting (le modèle continue de devenir meilleur pour prédire les images qu’il connait mais ne s’améliore pas sur ce qu’il ne connait pas).

Ensuite nous avons utilisé ce modèle sur les 24 composantes des handcrafted feature que nous avons fait précédemment (PCA, zones, edge). On compile et entraine le modèle et on obtient :



Cette fois ci nous avons une précision de 95,83% et une perte de 14,28% après 100 epochs, la précision est donc similaire entre les deux tests mais on voit que les pertes sont bien meilleures dans ce cas-là lorsqu'on utilise le modèle sur les données traitées (PCA, zones, edge) que sur les données brutes. De plus on remarque qu'ici même à l'epochs 100 nous n'avons pas d'overfitting (les pertes sont les mêmes que les données soient connues ou non par le modèle).

Conclusion :

Dans ce projet, nous avons mis en œuvre plusieurs techniques de machine learning afin de reconnaître automatiquement des chiffres manuscrits à partir d'images en 8×8 pixels.

La démarche était la suivante : préparation des données, réduction de dimension (ACP), extraction de features manuelles (zones, contours), création de pipelines, puis entraînement et évaluation de plusieurs modèles.

Deux approches principales ont été comparées :

- Le modèle SVC (Support Vector Classifier), optimisé avec GridSearchCV, a atteint une précision de 99,3% sur les données test, confirmant l'efficacité des SVM pour ce type de données vectorielles.
- Le modèle réseau de neurones (Sequential) a obtenu environ 95,8% de précision, avec une perte significativement réduite lorsqu'on utilise les features extraites (PCA + zonal + Sobel), démontrant l'importance d'un bon prétraitement.