Institut Polytechnique de l'Université de Nice

Departement Mathématiques Appliquées et Modélisation (MAM)

Introduction au Machine Learning

Amiel Metier
Hedi Chennoufi
Lucas Fert

Supervisor: Jean-Luc Bouchot, Mahmoud Elsawy

Rapport présenté en vue de satisfaire aux exigences de l'Université de Nice
pour l'obtention du diplôme de
Master of Science en Mathématiques Appliquées et Modélisation

June 13, 2025

Contents

1 Introduction

1.1 Présentation du Machine Learning

1.1.1 Définition et applications

Le Machine Learning est un champ d'étude de l'intelligence artificielle qui vise à donner aux machines la capacité d'« apprendre » à partir de données, via des modèles mathématiques.

1.1.2 Exemples d'applications

- Véhicules autonomes : Détection d'objets et adaptation en temps réel.
- Systèmes de recommandation : Netflix, Spotify analysent les préférences des utilisateurs.
- Détection de fraudes bancaires : Identification des transactions suspectes.

—

2 Exploration et Prétraitement des Données

2.1 Chargement et Visualisation

2.1.1 Présentation du dataset

Le Digits dataset de sklearn comprend 1 797 images de chiffres manuscrits de 0 à 9 repartie en 10 classes, où chaque image est une matrice de 8x8 pixels (en niveaux de gris, valeurs entre 0 et 16).

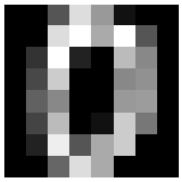

Figure 1: Chiffre 0 manuscrit

2.1.2 Visualisation des données

Nous allons d'abord commencer par observer les différents chiffres pour trouver des ressemblances entre eux et pour comprendre comment les distinguer.

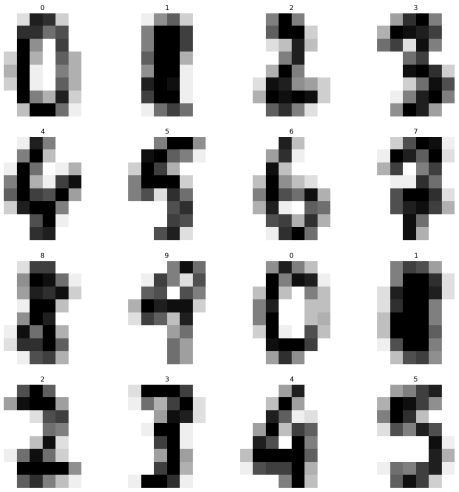

Figure 2: Exemples de chiffres manuscrits du dataset

Chaque case représente un chiffre manuscrit vennant du dataset. On observe des variations dans la forme des chiffres, ce qui peut poser problème pour la classification. Certains chiffres peuvent être difficiles à distinguer (par exemple, 8 et 3 ont des structures visuellement proches).

Pour mieux visualiser ces relations, on va plot une projection en 2D de la base de données via ACP (Analyse en composante principale).

Fonctionnement de l'ACP :

Soit $\mathbf{X} \in \mathbb{R}^{n \times d}$ la matrice des données contenant n images et d composantes.

I. **Centrage des données** :

$$\mathbf{X}_c = \mathbf{X} - \bar{\mathbf{X}}, \quad \text{où } \bar{\mathbf{X}} \text{ est la moyenne de chaque colonne}$$

II. **Calcul de la matrice de covariance** :

$$\mathbf{C} = \frac{1}{n-1} \mathbf{X}_c^\top \mathbf{X}_c$$

III. **Décomposition en valeurs propres** :

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i, \quad i = 1, \ldots, d$$

où λ_i sont les valeurs propres, et $\mathbf{v}_i$ les vecteurs propres associés.

IV. **Tri des valeurs propres (et vecteurs associés)** :

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$$

On construit la matrice des k premiers vecteurs propres associés aux plus grandes valeurs propres :

$$\mathbf{V}_k = [\mathbf{v}_1 \ \mathbf{v}_2 \ \ldots \ \mathbf{v}_k]$$

La première colonne de la matrice réorganisée est une composante qui capture la plus grande variance.

V. **Projection des données sur le nouvel espace** :

$$\mathbf{X}_{\text{réduit}} = \mathbf{X}_c \cdot \mathbf{V}_k$$

Ainsi, $\mathbf{X}_{\text{réduit}} \in \mathbb{R}^{n \times k}$ représente les données projetées sur les k composantes principales.

Si `n_components` $= 2$, alors la matrice initiale $\mathbf{X} \in \mathbb{R}^{1797 \times 64}$ est projetée sur un espace de dimension réduite :

$$\mathbf{X}_{\text{réduit}} \in \mathbb{R}^{1797 \times 2}$$

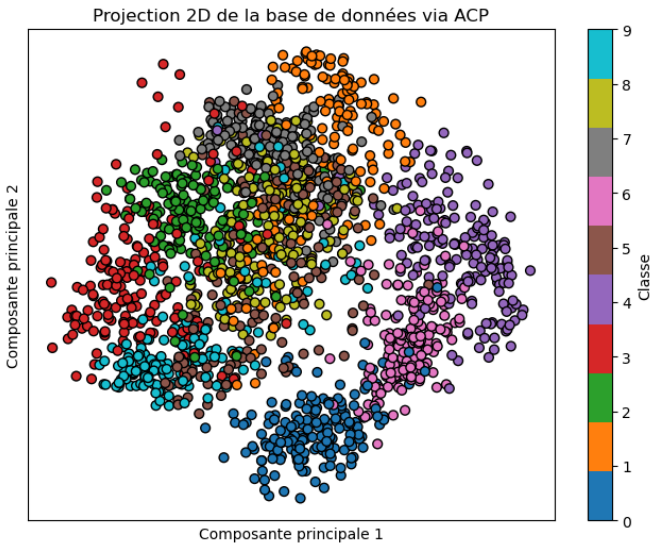

Figure 3: Projection en 2D de la base de donnée via ACP

Chaque point représente une image de chiffre. L'ACP réduit ces 64 dimensions à seulement ncomponents ici 2 (passé en paramètre) dimensions, tout en conservant autant que possible la variance.

Les groupes colorés montre ressemblance entre les chiffres. Certaines classes sont naturellement séparables (par ex. : les 0 ou les 1 forment des groupes distincts).

2.2 Prétraitement des Données

2.2.1 Normalisation

Les pixels sont normalisés dans [0,1] avec MinMaxScaler pour homogénéiser les valeurs. (On peut aussi les normaliser en les divisant par 16)

$$Formule : x_{\text{normé}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

2.2.2 Réduction de dimension (ACP)

Le graphique ci-dessous montre la proportion de variance expliquée par les composantes principales.

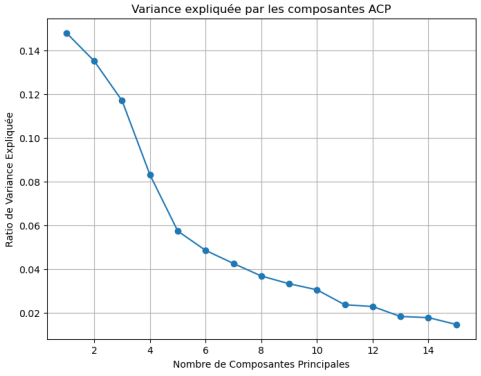

Figure 4: Variance expliquée en fonction du nombre de composantes principales

Les premières composantes capturent l'essentiel de l'information. À partir de 10, la courbe commence a ne plus trop diminué donc l'ajout de nouvelles dimensions n'améliore plus assez la représentation des chiffres.

3 Extraction et Transformation des Caractéristiques

3.1 Filtre Sobel : Détection de contours

Le filtre Sobel est utilisé pour détecter les contours des chiffres.

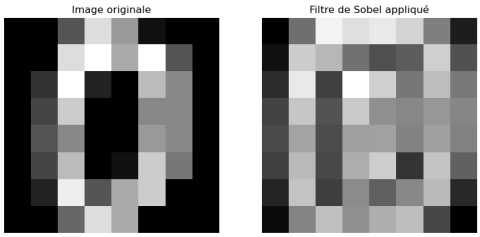

Figure 5: Détection des contours avec le filtre Sobel

On peut voir sur l'image de gauche l'image originale du chiffre 0 et sur celle de droite l'image du 0 après application du filtre Sobel.
Le filtre Sobel permet de calculer des gradients, donc de repérer où l'intensité des pixels varie brusquement : ce qui correspond aux bords des chiffres. Il met donc en évidence les zones de transition, aidant à différencier les chiffres visuellement similaires (comme par exemple 3 et 8) et il permet d'extraire des caractéristiques essentielles de chaques chiffres avant l'entraînement du modèle.

4 Modélisation et Entraînement

4.1 Séparation des Données

Les données sont séparées en: 80% entraînement et 20% test.

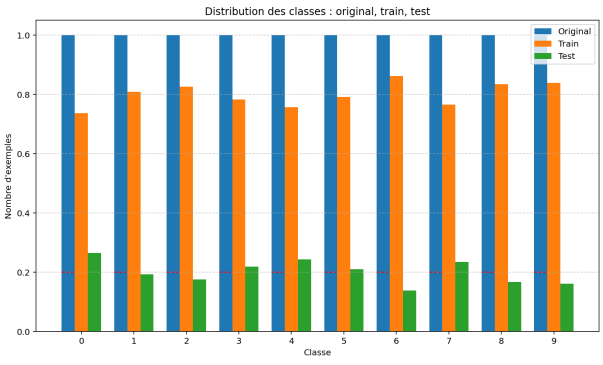

Figure 6: Distribution des classes avant et après la séparation train/test

On constate que chaque classe contient un nombre similaire d'exemples, évitant les déséquilibres dans l'apprentissage lors de l'entraînement.

5 Modèle SVM

5.1 Principe du SVM

Support Vector Machine est un classificateur supervisé efficace pour les problèmes de classification.

5.2 Noyaux testés

Les noyaux permettent de transformer les données afin de les rendre séparables dans un espace de dimension supérieure. On a utilisé 2 type de noyaux.

5.2.1 Noyau Linéaire

Le noyau linéaire applique une séparation linéaire entre les classes. Il est efficace lorsque les données sont naturellement séparables par un hyperplan, mais peut être limité pour des cas plus complexes où les données ne sont pas linéairement séparables.

5.2.2 Noyau RBF (Radial Basis Function)

Le noyau RBF projette les données dans un espace de dimension supérieure, permettant une meilleure séparation dans les cas où les classes ne sont pas linéairement séparables. Il repose sur une notion de proximité, les points proches ont une forte similarité, tandis que ceux éloignés ont une similarité proche de zéro. Par la suite nous avons priviligié d'utiliser le noyau RBF car il nous donnait de meilleurs resultats (0,99 de precison avec le noyau RBF contre 0,98 avec le noyau linéaire).

5.3 Hyperparamètres principaux

Les hyperparamètres influencent les performances du modèle et doivent être soigneusement ajustés :

5.3.1 Paramètre C

Le paramètre C contrôle l'équilibre entre maximisation de la marge et minimisation des erreurs de classification :

- **Valeur élevée de** C : Le modèle pénalise fortement les erreurs, ce qui le rend plus précis mais peut le rendre sensible au bruit.
- **Valeur faible de** C : Il favorise une marge plus large et une meilleure généralisation, mais peut tolérer quelques erreurs de classification.

5.3.2 Paramètre γ

Le paramètre γ influence la portée du noyau RBF :

- **Valeur élevée de** γ : Chaque point d'entraînement a une grande influence, ce qui peut mener à un surajustement où le modèle devient trop spécifique aux données d'entraînement.
- **Valeur faible de** γ : La généralisation est meilleure, mais les limites de décision peuvent être trop rigides et ne pas capturer la structure réelle des données.

5.3.3 Paramètre Réduction de dimension

La réduction de dimension vise à diminuer le nombre de variables dans un jeu de données tout en conservant l'information la plus pertinente

5.3.4 Paramètres sélectionnés

Les hyperparamètres testés pour l'optimisation du modèle sont :

- **Réduction de dimension (PCA)** : n_components $\in \{10, 20, 30\}$
- **Régularisation** : $C \in \{0.1, 1, 10\}$
- **Paramètre γ (noyau RBF)** : $\gamma \in \{\text{scale}, \text{auto}, 0.01, 0.001\}$

5.4 Stratégies Multi-classes

Les SVM sont initialement conçus pour la classification binaire, mais ils peuvent être étendus aux problèmes multi-classes à l'aide des stratégies suivantes :

5.4.1 OvR (One-vs-Rest)

La méthode One-vs-Rest consiste à entraîner n modèles, où chaque modèle apprend à distinguer une classe des autres. Lors de la prédiction, le modèle qui donne la plus grande probabilité est sélectionné.Cette approche est efficace mais peut être impactée par un déséquilibre des classes.

5.4.2 OvO (One-vs-One)

La méthode One-vs-One entraîne un classificateur pour chaque paire de classes. Avec n classes, il faut entraîner $\frac{n(n-1)}{2}$ modèles(donc dans notre cas pour 10 classes,nous devons entrainer 45 modèles). Lors de la prédiction, chaque modèle vote pour une classe, et la classe ayant reçu le plus de votes est attribuée. Cette approche est plus coûteuse en calcul mais peut offrir de meilleures performances lorsque les classes sont bien séparées.

5.5 Comparaison OvO vs OvR

On va calculer le score de chacun ainsi que le temps d'exécution pour ces deux méthodes. Voici ce que la console retourne :

```
 One-vs-One (OvO) Classification:
- Test score: 0.9916666666666667
- Number of classifiers trained: 17
ovo time : 0.1189s
- Impact: Suitable for small datasets but increases complexity.

 Question: How does OvO compare to OvR in execution time?
Try timing both methods and analyzing efficiency.

 One-vs-Rest (OvR) Classification:
- Test score: 0.9861111111111112
- Number of classifiers trained: 18
ovr time : 0.1399s
- Impact: Better for large datasets but less optimal for highly imbalanced data.

 Question: When would OvR be better than OvO?
Analyze different datasets and choose the best approach!
```

Figure 7: Affichage dans la console permettant de comparer OvO et OvR

On constate que pour OvO, le score de test est plus élevé et le temps d'exécution est plus court. Cela montre que, pour ce type de données, la méthode OvO est plus efficace. Cependant, lorsqu'on augmente la taille du jeu de données, on observe une inversion des performances : OvR devient plus rapide et obtient un meilleur score de test.

6 Optimisation et Validation

6.1 Matrice de confusion

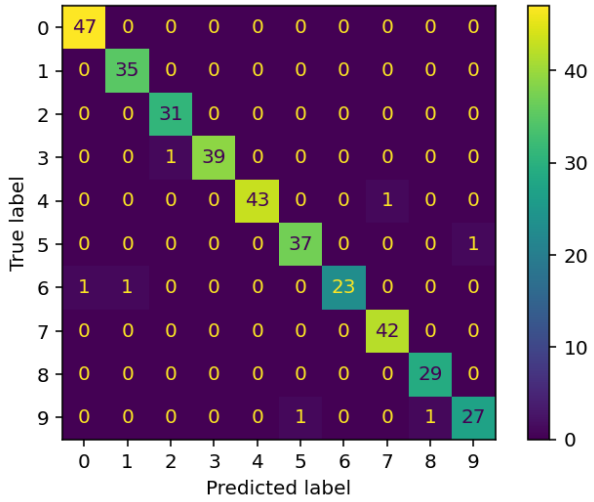

Figure 8: Matrice de confusion du meilleur modèle SVM

Les valeurs sur la diagonale montrent les prédictions correctes tandis que les valeurs hors diagonale correspondent aux erreurs du modèle. On constate des erreurs comme par exemple le 5 qui est confondu avec le 9 ou encore le 9 qui est confondu avec le 8 mais sur 360 predictions, il n'a eu que 7 erreurs ,ce qui est un très bon ratio.

7 Réseau de neuronnes

Un réseau de neurones artificiel est un modèle d'intelligence artificielle qui essaye de répéter le fonctionnement du cerveau humain. Il est composé de plusieurs couches de neurones connectés entre elles qui traitent les données et apprennent à reconnaître des motifs ou bien à prendre des décisions.
Voici le code que nous avons implémenté pour :

```
digits = load_digits()
X = digits.data
y = digits.target
scaler = MinMaxScaler()
X = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42, stratify=y)
model = keras.Sequential([keras.layers.Input(shape=(64,)), keras.layers.
   Dense(32, activation='relu'),keras.layers.Dense(10, activation='
   softmax')])
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
   metrics=['accuracy'])
history = model.fit(X_train, y_train,validation_split=0.2,epochs=30,
   batch_size=32,verbose=1)
```

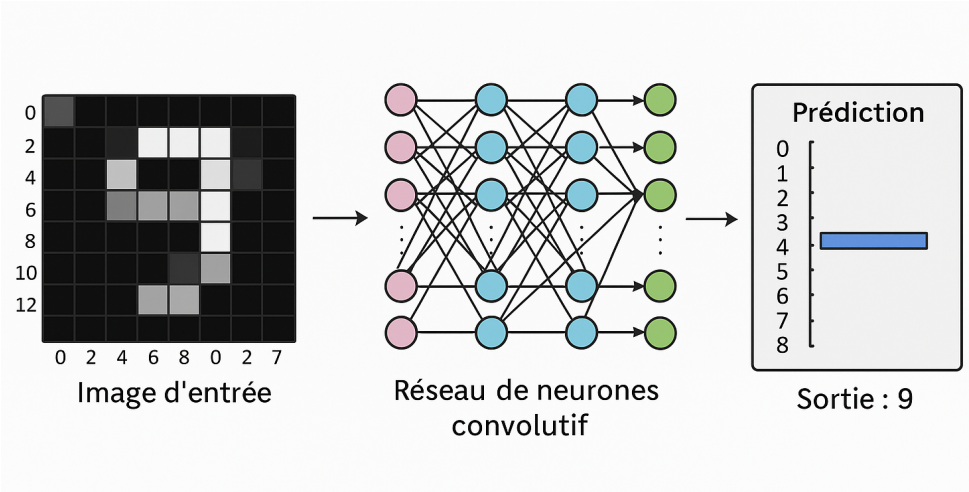

Figure 9: schéma explicatif du réseau de neuronnes

Nous avons utilisé des images en niveaux de gris sous forme de matrice de pixels. Les neurones analysent les pixels et extraient des caractéristiques importantes (traits, formes, courbes). Après plusieurs étapes de transformation et de calcul, le réseau prédit le chiffre affiché dans l'image. Ici, le chiffre predit est 9.

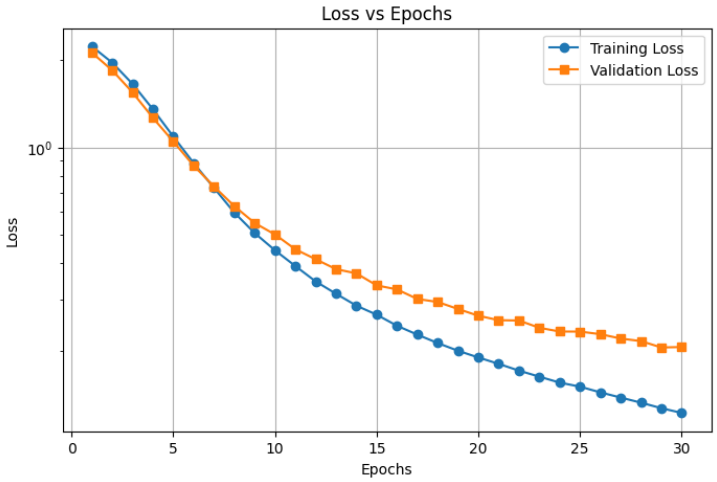

Figure 10: Exemple de prédiction de notre IA

Ce graphique montre que le modèle apprend efficacement au fil des étapes, la perte d'entraînement diminue de manière régulière, nous montrant que le modèle s'ajuste bien aux données. La perte de la validation diminue également de façon progressive. On en déduit que le modèle arrive bien à généraliser sur des données non vues. L'absence de remontée de la courbe de validation indiquent qu'il n'y a pas de surapprentissage. On peut donc conclure que l'entraînement est bien et que les performances du modèle sont satisfaisantes jusqu'à ce stade.

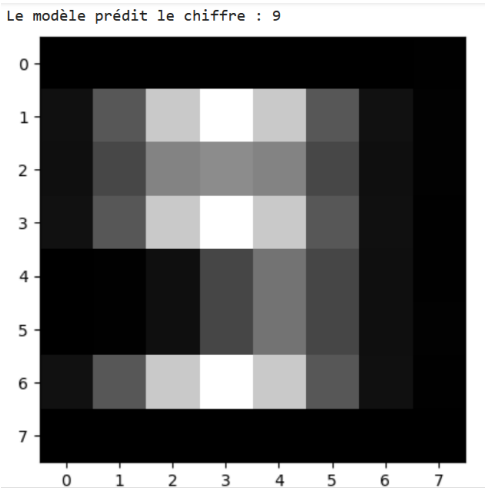

Figure 11: Exemple de prédiction de notre IA

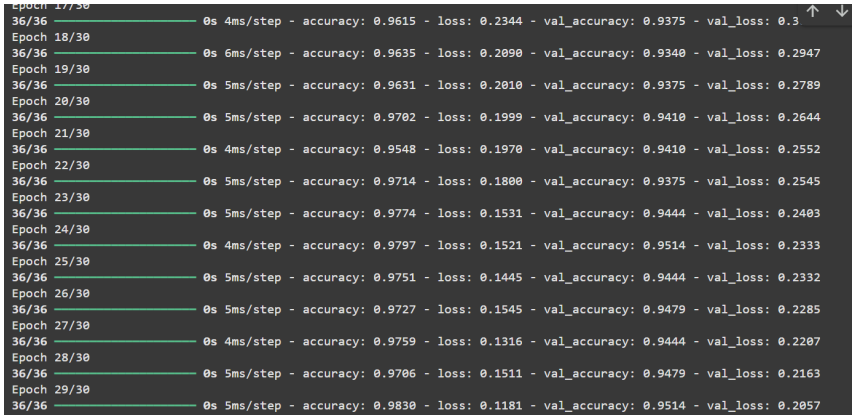

Figure 12: Fiabilité du test

Nous pouvons voir que notre IA arrive à retouvrer correctement les chiffres d'après notre exemples et ceux d'en plus de 95% des cas.
En faisant du pré-traitement sur nos données le score était plus faible donc inutile dans notre cas.

8 Résultats et Conclusion

8.1 Performance du modèle

8.1.1 Précision obtenue

Le modèle le plus optimisé atteint une précision de 99%, c'est celui basé sur SVC avec un kernel dit rbf.
Pour l'optimiser on a pris comme paramètre, 20 compenants pour le PCA, un C=10 et un gamma égal à 0.001 et bien sur on a utilisé sobel et la matrice de zone.
Les autres modèles ayant des bons pourcentages mais pas autant que celui là.

8.1.2 Comparaison des stratégies

Méthode	Performance	Temps d'exécution
PCA + SVM 'linear'	Bonne	Rapide
PCA + SVM 'rbf'	Excellente	Légèrement plus lent
OvO	Très précis	Long
OvR	Plus rapide	Moins précis sur petits datasets
Réseau de neurones	correct	depend

Table 1: Comparaison des différentes stratégies de classification