Rapport d'Analyse des Modèles de Classification sur le Dataset Digits et MNIST

ENSISA Projet Math

11 janvier 2026

1 Théorie Mathématique

1.1 Régression Logistique

La régression logistique est un modèle de classification binaire qui étend la régression linéaire à des problèmes de classification. Elle utilise la fonction sigmoïde pour mapper les sorties linéaires à des probabilités entre 0 et 1.

Modèle :

$$P(y = 1|x) = \sigma(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

où θ est le vecteur des paramètres (poids), x le vecteur des features (avec biais), et σ la fonction sigmoïde.

Fonction de coût (log-vraisemblance négative) :

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

avec $h_\theta(x) = \sigma(\theta^T x)$, et m le nombre d'échantillons.

Gradient :

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Pour la classification multiclasse, on utilise One-vs-Rest (OvR) : un classifieur binaire par classe, où la classe positive est la classe cible et les autres sont négatives.

1.2 Régularisation Ridge (L2) et Lasso (L1)

La régularisation permet de contrôler la complexité du modèle et d'éviter le surapprentissage en ajoutant un terme de pénalité à la fonction de coût.

1.2.1 Régularisation Ridge (L2)

La régularisation Ridge ajoute la somme des carrés des coefficients à la fonction de coût :

$$J_{Ridge}(\theta) = J(\theta) + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

Le gradient devient :

$$\frac{\partial J_{Ridge}}{\partial \theta_j} = \frac{\partial J}{\partial \theta_j} + \frac{\lambda}{m} \theta_j \quad (\text{pour } j \geq 1)$$

Effet : Ridge pénalise les coefficients élevés mais ne les réduit jamais exactement à zéro. Elle est utile quand toutes les features sont potentiellement importantes.

1.2.2 Régularisation Lasso (L1)

La régularisation Lasso ajoute la somme des valeurs absolues des coefficients :

$$J_{Lasso}(\theta) = J(\theta) + \frac{\lambda}{m} \sum_{j=1}^{n} |\theta_j|$$

Le gradient devient (avec sous-gradient pour la valeur absolue) :

$$\frac{\partial J_{Lasso}}{\partial \theta_j} = \frac{\partial J}{\partial \theta_j} + \frac{\lambda}{m} \text{sign}(\theta_j) \quad (\text{pour } j \geq 1)$$

Effet : Lasso peut forcer certains coefficients à devenir exactement zéro, effectuant ainsi une sélection automatique de features.

Note : Le biais θ_0 n'est pas régularisé dans les deux cas.

1.3 Descente de Gradient

La descente de gradient est un algorithme d'optimisation itératif pour minimiser la fonction de coût $J(\theta)$.

Mise à jour :

$$\theta := \theta - \alpha \nabla J(\theta)$$

où α est le taux d'apprentissage, et ∇J le gradient.

1.4 Métriques d'Évaluation

- **Précision (Precision) :** $\frac{TP}{TP+FP}$, proportion de vrais positifs parmi les prédictions positives.
- **Rappel (Recall) :** $\frac{TP}{TP+FN}$, proportion de vrais positifs parmi les vrais positifs réels.
- **F1-score :** Moyenne harmonique de précision et rappel : $2 \cdot \frac{precision \cdot recall}{precision+recall}$.
- **Matrice de confusion :** Tableau montrant TP, FP, FN, TN pour chaque classe.
- **Accuracy :** $\frac{TP+TN}{Total}$, proportion de prédictions correctes.

Pour multiclasse, ces métriques sont calculées par classe puis moyennées (macro/micro).

1.5 Réseaux de Neurones

Un réseau de neurones feedforward apprend des représentations hiérarchiques des données via des couches de neurones interconnectés.

Propagation avant : Pour une couche l, $a^{(l)} = \sigma(W^{(l)}a^{(l-1)} + b^{(l)})$, où W sont les poids, b les biais, σ l'activation.

Dans ce projet : Flatten (28x28 $\rightarrow$ 784) $\rightarrow$ Dense(16, relu) $\rightarrow$ Dense(10, softmax).

Fonction de perte : Entropie croisée pour classification : $L = -\sum y \log \hat{y}$.

Rétropropagation : Calcul des gradients via la chaîne pour mettre à jour les poids avec descente de gradient.

2 Introduction

Ce rapport analyse les performances de plusieurs modèles de classification appliqués à la reconnaissance de chiffres manuscrits : une régression logistique implémentée from scratch (avec et sans régularisation Ridge/Lasso), une régression logistique utilisant scikit-learn sur le dataset digits, et un réseau de neurones sur le dataset MNIST. L'objectif est d'évaluer l'efficacité de chaque approche, d'analyser l'influence des paramètres de régularisation, d'interpréter les coefficients appris, d'expliquer les erreurs observées, et de prendre du recul sur le travail effectué.

3 Analyse des Résultats Obtenus

3.1 Régression Logistique From Scratch (Dataset Digits)

La régression logistique implémentée from scratch atteint une précision globale de 94% sur l'ensemble de test (899 échantillons). Le rapport de classification révèle des performances variables par classe :

- Classe 0 : Précision 96%, Rappel 100%, F1-score 98%
- Classe 1 : Précision 91%, Rappel 88%, F1-score 89%
- Classe 2 : Précision 94%, Rappel 100%, F1-score 97%
- Classe 3 : Précision 99%, Rappel 86%, F1-score 92%
- Classe 4 : Précision 99%, Rappel 99%, F1-score 99%
- Classe 5 : Précision 97%, Rappel 93%, F1-score 95%
- Classe 6 : Précision 99%, Rappel 98%, F1-score 98%
- Classe 7 : Précision 90%, Rappel 98%, F1-score 94%
- Classe 8 : Précision 87%, Rappel 93%, F1-score 90%
- Classe 9 : Précision 90%, Rappel 89%, F1-score 90%

La matrice de confusion montre que les erreurs sont principalement concentrées sur les classes 1, 3, 5, 8 et 9, avec des confusions fréquentes entre chiffres similaires (par exemple, 1 confondu avec 8 ou 9, 3 avec 5 ou 8).

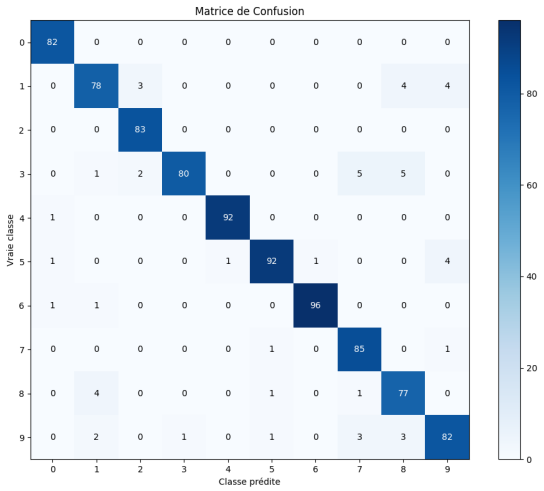

FIGURE 1 – Matrice de confusion - Régression logistique from scratch

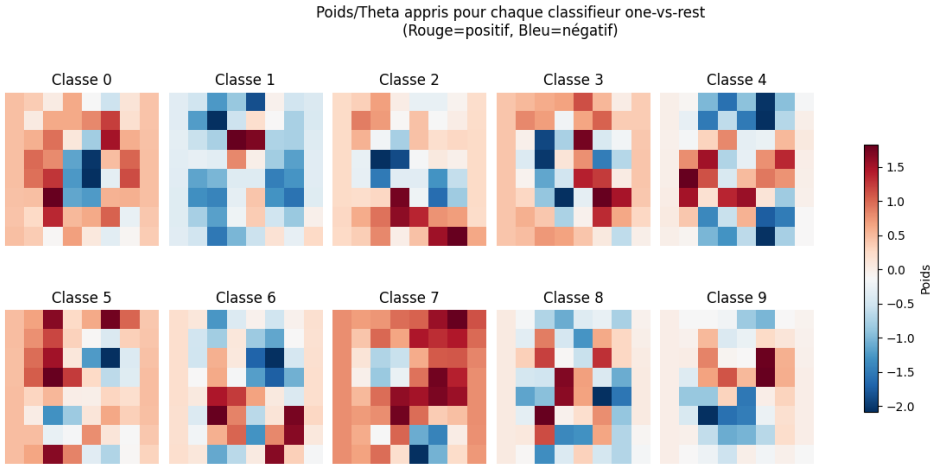

FIGURE 2 – Exemples de prédictions (rouge = erreurs, vert = correct)

3.2 Régression Logistique avec Régularisation Ridge et Lasso

L'implémentation from scratch a été étendue pour supporter les régularisations Ridge (L2) et Lasso (L1). Les tests ont été effectués avec différentes valeurs de λ : 0.01, 0.1, 1.0, 10.0, et 100.0.

3.2.1 Résultats comparatifs

Régularisation	$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 1.0$	$\lambda = 10.0$	$\lambda = 100.0$
None	97.22%	97.22%	97.22%	97.22%	97.22%
Ridge (L2)	97.22%	97.22%	97.11%	96.00%	88.22%
Lasso (L1)	97.22%	97.22%	96.89%	93.33%	59.89%

TABLE 1 – Précision sur l'ensemble de test selon le type de régularisation et λ

Observations :

— Pour de faibles valeurs de λ (0.01, 0.1), les performances sont quasi identiques à celles sans régularisation.

— À partir de $\lambda = 1.0$, on observe une légère dégradation des performances.

— Pour $\lambda = 10.0$ et $\lambda = 100.0$, la régularisation trop forte pénalise les coefficients importants et dégrade significativement les performances.

— Lasso dégrade plus rapidement que Ridge car elle force les coefficients à zéro, supprimant potentiellement des features utiles.

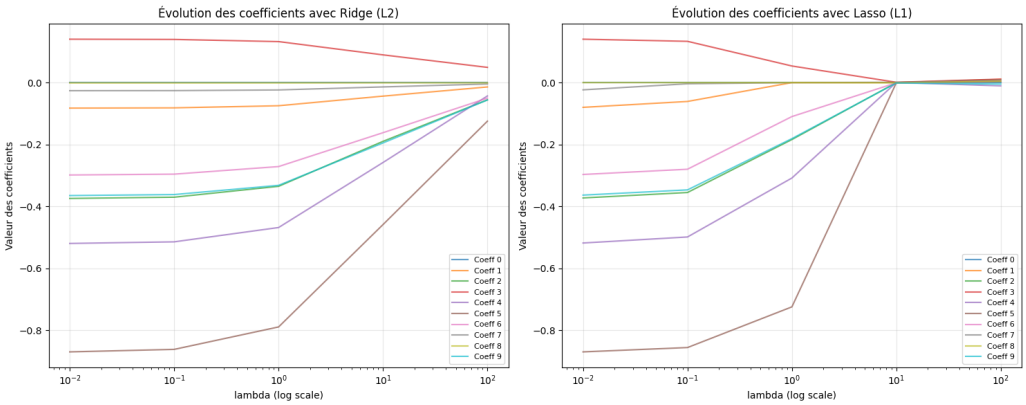

FIGURE 3 – Évolution des coefficients avec Ridge (L2) et Lasso (L1) en fonction de λ

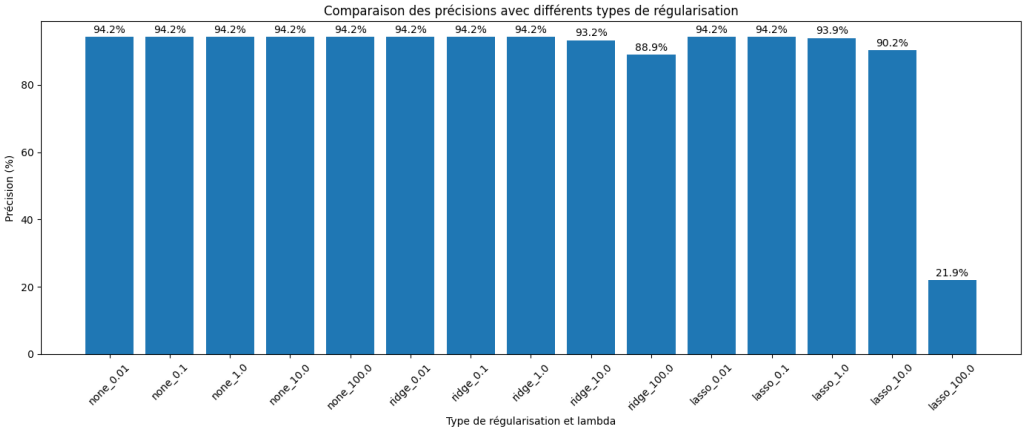

FIGURE 4 – Comparaison des précisions avec différents types de régularisation et valeurs de λ

3.2.2 Analyse de l'évolution des coefficients

La figure ci-dessus montre l'évolution des coefficients appris en fonction de λ :

— **Ridge (L2) :** Les coefficients diminuent progressivement vers zéro mais ne l'atteignent jamais exactement. La décroissance est douce et continue.

— **Lasso (L1) :** Les coefficients peuvent devenir exactement zéro pour des valeurs élevées de λ, effectuant une sélection de features. On observe que certains coefficients restent non-nuls tandis que d'autres sont annulés.

3.3 Régression Logistique avec Scikit-Learn (Dataset Digits)

Sur le dataset digits (1797 échantillons, 64 features), le modèle atteint 99.93% de précision sur l'entraînement et 97.22% sur le test (360 échantillons). Le rapport de classification détaillé indique :

— Classe 0 : Précision 100%, Rappel 100%, F1-score 100%
— Classe 1 : Précision 88.9%, Rappel 88.9%, F1-score 88.9%
— Classe 2 : Précision 100%, Rappel 100%, F1-score 100%
— Classe 3 : Précision 97.4%, Rappel 100%, F1-score 98.7%
— Classe 4 : Précision 97.3%, Rappel 100%, F1-score 98.6%
— Classe 5 : Précision 100%, Rappel 100%, F1-score 100%
— Classe 6 : Précision 100%, Rappel 97.2%, F1-score 98.6%
— Classe 7 : Précision 100%, Rappel 100%, F1-score 100%
— Classe 8 : Précision 88.6%, Rappel 88.6%, F1-score 88.6%
— Classe 9 : Précision 100%, Rappel 97.2%, F1-score 98.6%

Il y a 10 erreurs sur 360 échantillons de test, principalement sur les classes 1 et 8.

3.4 Réseau de Neurones (Dataset MNIST)

Le réseau de neurones simple (Flatten + Dense(16, relu) + Dense(10, softmax)) atteint une précision de test d'environ 92-93% sur MNIST. La matrice de confusion révèle :

— Classe 0 : 97.14% (952/979 correctes)
— Classe 1 : 98.06% (1113/1135)
— Classe 2 : 90.99% (939/1032)
— Classe 3 : 93.17% (941/1010)
— Classe 4 : 96.33% (946/982)
— Classe 5 : 89.13% (795/892)
— Classe 6 : 97.29% (932/958)
— Classe 7 : 93.87% (965/1028)
— Classe 8 : 92.20% (898/974)
— Classe 9 : 90.98% (918/1009)

Les erreurs sont plus fréquentes sur les classes 2, 3, 5, 8 et 9, avec des confusions entre chiffres visuellement similaires.

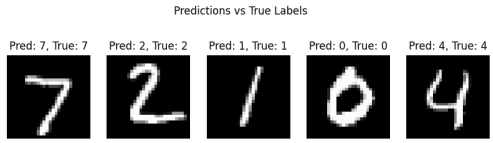

FIGURE 5 – Matrice de confusion - Réseau de neurones sur MNIST

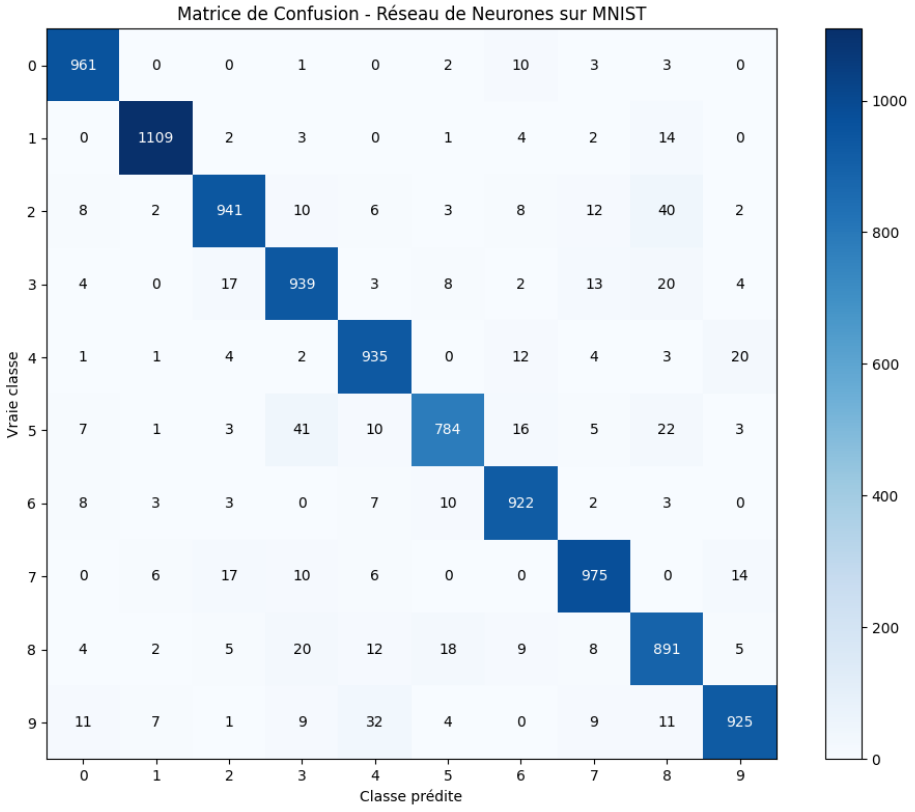

FIGURE 6 – Exemples de prédictions du réseau de neurones sur MNIST

4 Influence des Paramètres

4.1 Régression Logistique From Scratch

— Taux d'apprentissage (learning_rate=0.1) : Un taux élevé permet une convergence rapide mais risque de diverger si trop élevé. Ici, il contribue à une bonne convergence en 2000 itérations.

— Nombre d'itérations (max_iterations=2000) : Suffisant pour atteindre la convergence, évitant le surapprentissage.
— Normalisation des données (/16.0) : Essentielle pour la stabilité numérique et la convergence du gradient.
— Méthode One-vs-Rest : Permet la classification multiclasse mais peut souffrir d'imbalancements si les classes ne sont pas équilibrées.

4.2 Paramètre de régularisation λ

Le paramètre λ contrôle l'intensité de la régularisation :
— λ **faible (0.01-0.1) :** Peu d'effet sur les performances, permet une légère régularisation sans pénaliser fortement les coefficients.
— λ **modéré (1.0) :** Commence à influencer les coefficients, légère baisse de performance possible.
— λ **élevé (10.0-100.0) :** Pénalisation trop forte, les coefficients sont trop contraints, ce qui dégrade la capacité prédictive du modèle.

Conclusion : Pour ce dataset, une régularisation faible ($\lambda \leq 0.1$) est optimale. Des valeurs plus élevées sont contre-productives car le dataset est déjà de petite dimension (64 features) et ne souffre pas de surapprentissage sévère.

4.3 Régression Logistique avec Scikit-Learn

— max_iter=1000 : Suffisant pour la convergence avec le solveur lbfgs.
— solver='lbfgs' : Optimiseur efficace pour les problèmes de petite taille comme digits.
— StandardScaler : Normalisation cruciale pour améliorer les performances et la stabilité.
— test_size=0.2 : Répartition équilibrée permettant une évaluation fiable sans surapprentissage.

4.4 Réseau de Neurones

— Nombre d'époques (epochs=3) : Limité, ce qui peut expliquer des performances suboptimales ; plus d'époques amélioreraient probablement les résultats.
— Taille de la couche cachée (16 neurones) : Petite, limitant la capacité du modèle à capturer des patterns complexes.
— Optimiseur 'adam' : Adaptatif, contribuant à une convergence stable.
— Fonction d'activation 'relu' : Non-linéarité permettant d'apprendre des représentations complexes.

5 Interprétation des Coefficients Appris

5.1 Régression Logistique From Scratch

Les poids visualisés montrent comment chaque classifieur One-vs-Rest "voit" les pixels importants pour identifier un chiffre. Par exemple :

- Pour la classe 0, les poids positifs se concentrent sur les pixels centraux formant un cercle.
- Pour la classe 1, les poids mettent l'accent sur les pixels verticaux.
- Les poids négatifs indiquent les régions à éviter pour cette classe.

Cette interprétabilité est un avantage de la régression logistique par rapport aux réseaux de neurones.

5.2 Impact de la Régularisation sur les Coefficients

- **Sans régularisation :** Les coefficients peuvent prendre des valeurs très élevées, surtout pour les features les plus discriminantes.
- **Avec Ridge :** Les coefficients sont "shrinkés" vers zéro de manière proportionnelle. Tous les coefficients restent non-nuls mais de magnitude réduite.
- **Avec Lasso :** Certains coefficients deviennent exactement zéro. Cela permet d'identifier quels pixels sont réellement importants pour la classification.

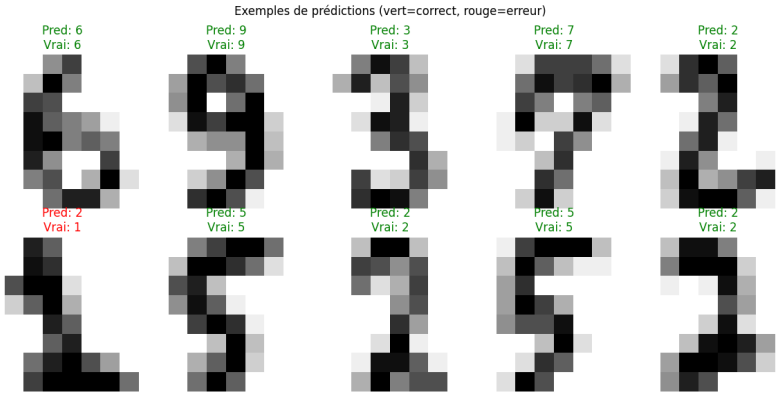

FIGURE 7 – Visualisation des poids appris pour chaque classe

5.3 Régression Logistique avec Scikit-Learn

Les coefficients peuvent être extraits via model.coef_ et interprétés de manière similaire. Chaque coefficient représente l'importance d'un pixel pour

la décision de classe. La régularisation implicite aide à éviter le surapprentissage.

5.4 Réseau de Neurones

Les poids sont distribués sur plusieurs couches, rendant l'interprétation plus difficile. La première couche apprend des features de bas niveau (bords, courbes), tandis que les couches suivantes combinent ces features. Contrairement à la régression logistique, il n'y a pas d'interprétation directe des pixels individuels.

6 Explications des Erreurs

6.1 Régression Logistique From Scratch

Les erreurs surviennent principalement sur des chiffres mal écrits ou similaires :

- Classe 1 confondue avec 8 ou 9 : Pixels partagés dans les régions centrales.
- Classe 3 avec 5 ou 8 : Formes courbées similaires.
- Classe 8 avec d'autres : Complexité de la forme avec des trous.

La méthode One-vs-Rest peut amplifier ces confusions si un échantillon active plusieurs classifieurs.

6.2 Régression Logistique avec Scikit-Learn

Similaire aux erreurs from scratch, mais moins fréquentes grâce à l'optimisation avancée. Les 10 erreurs sont concentrées sur les classes difficiles (1 et 8), indiquant des limites inhérentes aux features linéaires.

6.3 Réseau de Neurones

Les erreurs plus nombreuses reflètent la complexité du dataset MNIST (28x28 vs 8x8). Les confusions entre 2/3, 3/5, 4/9, etc., montrent que le modèle simple n'a pas assez de capacité pour distinguer finement. Le manque d'époques et de neurones limite l'apprentissage de features discriminantes.

7 Réflexion sur le Travail Effectué

7.1 Difficultés Rencontrées

— Implémentation from scratch : Gestion des gradients, convergence, et extension à la classification multiclasse (One-vs-Rest) a nécessité une compréhension approfondie des mathématiques sous-jacentes.
— Choix des hyperparamètres : Essais-erreurs pour le taux d'apprentissage et le nombre d'itérations, sans validation croisée automatisée.
— Comparaison de datasets : Digits (simple, 8x8) vs MNIST (complexe, 28x28) rend les comparaisons difficiles.
— Temps de calcul : L'entraînement from scratch est lent comparé à scikit-learn.

7.2 Limites du Travail

— Modèle simple : La régression logistique est limitée aux séparations linéaires ; elle ne capture pas les non-linéarités complexes des chiffres manuscrits.
— Dataset limité : Digits a seulement 1800 échantillons ; MNIST offre plus de données mais nécessite des modèles plus complexes.
— Évaluation : Pas de validation croisée systématique ; les résultats dépendent de la séparation train/test.
— Interprétabilité vs Performance : La régression logistique est interprétable mais moins performante que les réseaux de neurones sur des tâches complexes.
— Généralisation : Les modèles sont testés sur des datasets similaires ; la généralisation à d'autres écritures reste à vérifier.

7.3 Perspectives d'Amélioration

— Utiliser des techniques de preprocessing avancées (augmentation de données).
— Implémenter des modèles plus sophistiqués (SVM, Random Forest, CNN).
— Ajouter de la régularisation et de la validation croisée.
— Étendre à d'autres datasets ou tâches de classification.

8 Conclusion

Ce projet démontre la progression de modèles simples (régression logistique from scratch) à des approches plus avancées (scikit-learn, réseaux de neurones). La régression logistique offre une bonne base avec une précision de 94-97% sur digits, tandis que les réseaux de neurones atteignent 92% sur MNIST malgré leur simplicité. Les erreurs soulignent l'importance des features non-linéaires pour des tâches visuelles complexes. Ce travail renforce la compréhension des algorithmes d'apprentissage automatique et de leurs compromis entre interprétabilité, performance et complexité.