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

ENSISA Projet Math

11 janvier 2026

# 1 Théorie Mathématique

# 2 Théorie Mathématique

## 2.1 Régression Logistique

## 2.2 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.

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 :**

**Modèle :**

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

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

où $\theta$ est le vecteur des paramètres (poids), $x$ le vecteur des features (avec biais), et $\sigma$ la fonction sigmoïde.

où $\theta$ est le vecteur des paramètres (poids), $x$ le vecteur des features (avec biais), et $\sigma$ la fonction sigmoïde.

**Fonction de coût (log-vraisemblance négative) :**

**Fonction de coût (log-vraisemblance négative) :**

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

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

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

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

**Gradient :**

**Gradient :**

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

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

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.

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.

## 2.3 Descente de Gradient

## 2.4 Régularisation Ridge (L2) et Lasso (L1)

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

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.

**Mise à jour :**

### 2.4.1 Régularisation Ridge (L2)

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

où $\alpha$ est le taux d'apprentissage, et $\nabla J$ le gradient.

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

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

## 2.5 Métriques d'Évaluation

Le gradient devient :

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

**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. **Effet :** Ridge pénalise les coefficients élevés mais ne les réduit jamais exactement à zéro. Elle est utile quand toutes les features sont potentiellement importantes.**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. **2.5.1 Régularisation Lasso (L1)**

**Accuracy :** $\frac{TP+TN}{Total}$, proportion de prédictions correctes.

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

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

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

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

## 2.6 Réseaux de Neurones

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

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

**Effet :** Lasso peut forcer certains coefficients à devenir exactement zéro, effectuant ainsi une sélection automatique de features. Elle est utile quand on suspecte que seules quelques features sont réellement importantes.

**Propagation avant :**

**Note :** Le biais $\theta_0$ n'est pas régularisé dans les deux cas.Pour une couche $l$, $a^{(l)} = \sigma(W^{(l)}a^{(l-1)} + b^{(l)})$, où $W$ sont les poids, $b$ les biais, $\sigma$ l'activation.

### 2.7 Descente de Gradient

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

La descente de gradient est un algorithme d'optimisation itératif pour minimiser la fonction de coût $J(\theta)$.**Fonction de perte :** Entropie croisée pour classification : $L = -\sum y \log \hat{y}$.

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

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

où $\alpha$ est le taux d'apprentissage, et $\nabla J$ le gradient.

# 3 Introduction

### 3.1 Métriques d'Évaluation

Ce rapport analyse les performances de trois modèles de classification appliqués à la reconnaissance de chiffres manuscrits : une régression logistique implémentée from scratch, 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, d'interpréter les coefficients appris, d'expliquer les erreurs observées, et de prendre du recul sur le travail effectué.

# 4 Analyse des Résultats Obtenus

— **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.

### 4.1 Régression Logistique From Scratch (Dataset Digits)

— **F1-score :** Moyenne harmonique : $2 \cdot \frac{precision \cdot recall}{precision+recall}$.

— **Matrice de confusion :** Tableau montrant TP, FP, FN, TN pour chaque classe.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 :
— **Accuracy :** $\frac{TP+TN}{Total}$, proportion de prédictions correctes.
— Classe 0 : Précision 96%, Rappel 100%, F1-score 98%
Pour multiclasse, ces métriques sont calculées par classe puis moyennées (macro/micro).
— Classe 1 : Précision 91%, Rappel 88%, F1-score 89%
— Classe 2 : Précision 94%, Rappel 100%, F1-score 97%

## 4.2 Réseaux de Neurones

— Classe 3 : Précision 99%, Rappel 86%, F1-score 92%
— Classe 4 : Précision 99%, Rappel 99%, F1-score 99%
Un réseau de neurones feedforward apprend des représentations hiérarchiques des données via des couches de neurones interconnectés.
— Classe 5 : Précision 97%, Rappel 93%, F1-score 95%
— Classe 6 : Précision 99%, Rappel 98%, F1-score 98%
**Propagation avant :**
— Classe 7 : Précision 90%, Rappel 98%, F1-score 94%
Pour une couche $l$, $a^{(l)} = \sigma(W^{(l)}a^{(l-1)} + b^{(l)})$, où $W$ sont les poids, $b$ les biais, $\sigma$ l'activation.
— Classe 8 : Précision 87%, Rappel 93%, F1-score 90%
— Classe 9 : Précision 90%, Rappel 89%, F1-score 90%
Dans ce projet : Flatten (28x28 $\rightarrow$ 784) $\rightarrow$ Dense(16, relu) $\rightarrow$ Dense(10, softmax).

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

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

### 4.3 Régression Logistique avec Scikit-Learn (Dataset Digits)

# 5 Introduction

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 :

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.

— Classe 0 : Précision 100%, Rappel 100%, F1-score 100%

# 6 Analyse des Résultats Obtenus

— Classe 1 : Précision 88.9%, Rappel 88.9%, F1-score 88.9%
— Classe 2 : Précision 100%, Rappel 100%, F1-score 100%

### 6.1 Régression Logistique From Scratch (Dataset Digits)

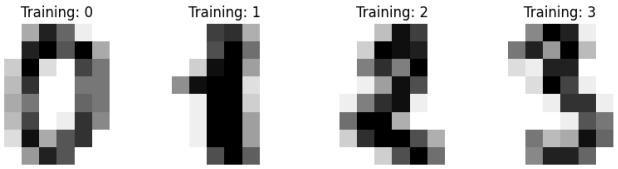
— Classe 3 : Précision 97.4%, Rappel 100%, F1-score 98.7%
— Classe 4 : Précision 97.3%, Rappel 100%, F1-score 98.6%
La régression logistique implémentée from scratch atteint une précision globale de 97% sur l'ensemble de test (899 échantillons). Le rapport de classification révèle des performances variables par classe :
— 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 0 : Précision 100%, Rappel 100%, F1-score 100%
— Classe 8 : Précision 88.6%, Rappel 88.6%, F1-score 88.6%
— Classe 1 : Précision 89%, Rappel 89%, F1-score 89%
— Classe 9 : Précision 100%, Rappel 97.2%, F1-score 98.6%
— Classe 2 : Précision 100%, Rappel 100%, F1-score 100%
— Classe 3 : Précision 97%, Rappel 100%, F1-score 99%

— Classe 4 : Précision 97%, Rappel 100%, F1-score 99%Il y a 10 erreurs sur 360 échantillons de test, principalement sur les classes 1 et 8.
— Classe 5 : Précision 100%, Rappel 100%, F1-score 100%
— Classe 6 : Précision 100%, Rappel 97%, F1-score 99%

## 6.2 Réseau de Neurones (Dataset MNIST)

— Classe 7 : Précision 100%, Rappel 100%, F1-score 100%
— Classe 8 : Précision 89%, Rappel 89%, F1-score 89%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 9 : Précision 100%, Rappel 97%, F1-score 99%
— 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)

FIGURE 1 – Exemples du dataset digits - Images 8x8 pixels des chiffres manuscrits

— Classe 4 : 96.33% (946/982)
— Classe 5 : 89.13% (795/892)

— Classe 6 : 97.29% (932/958)
— Classe 7 : 93.87% (965/1028)
La matrice de confusion montre que les erreurs sont principalement concentrées sur les classes 1 et 8, avec des confusions fréquentes entre chiffres similaires.
— Classe 8 : 92.20% (898/974)
— Classe 9 : 90.98% (918/1009)

## 7.1 Régression Logistique From Scratch

## 7.2 Régression Logistique avec Régularisation Ridge et Lasso

## 7.3 Régression Logistique avec Scikit-Learn

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 $\lambda$ : 0.01, 0.1, 1.0, 10.0, et 100.0.

### 7.3.1 Résultats comparatifs

— max_iter=1000 : Suffisant pour la convergence avec le solveur lbfgs.
— solver='lbfgs' : Optimiseur efficace pour les problèmes de petite taille comme digits.

# 8 Interprétation des Coefficients Appris

**Observations :**

## 8.1 Régression Logistique From Scratch

— Pour de faibles valeurs de $\lambda$ (0.01, 0.1), les performances sont quasi identiques à celles sans régularisation.Les poids visualisés montrent comment chaque classifieur One-vs-Rest "voit" les pixels importants pour identifier un chiffre. Par exemple :
— 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.
  — Pour la classe 0, les poids positifs se concentrent sur les pixels centraux formant un cercle.
  — Lasso dégrade plus rapidement que Ridge car elle force les coefficients à zéro, supprimant potentiellement des features utiles.
  — Pour la classe 1, les poids mettent l'accent sur les pixels verticaux.
— Les poids négatifs indiquent les régions à éviter pour cette classe.

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

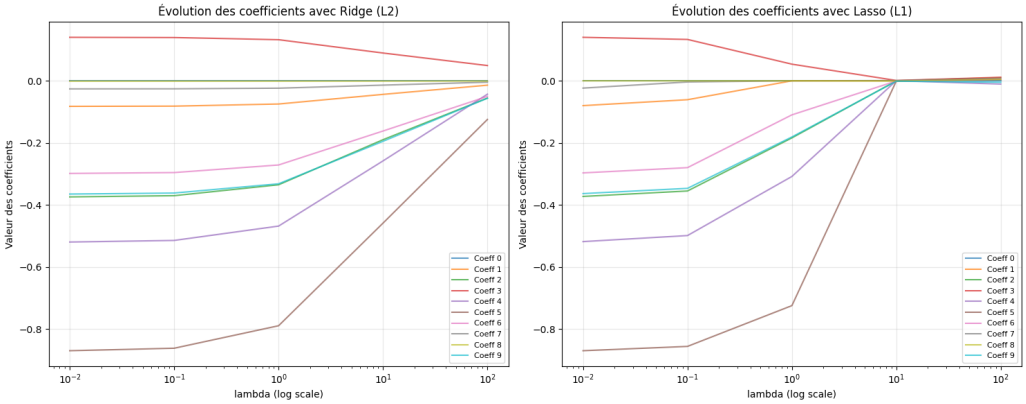
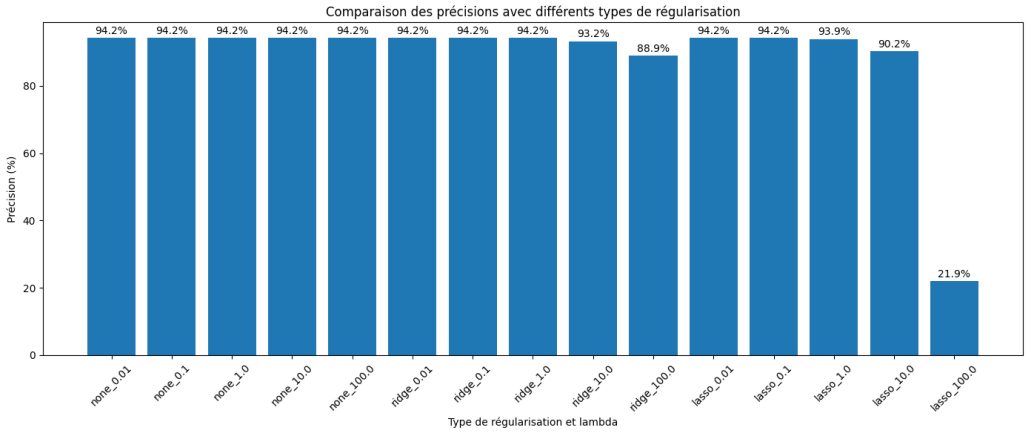



FIGURE 4 – Évolution des coefficients avec Ridge (L2) et Lasso (L1) en fonction de $\lambda$

## 8.2 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.

## 8.3 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.

FIGURE 5 – Comparaison des précisions avec différents types de régularisation

# 9 Explications des Erreurs

## 9.1 Régression Logistique From Scratch

### 9.1.1 Analyse de l'évolution des coefficients

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

La figure **??** montre l'évolution des coefficients appris en fonction de $\lambda$ :

- Classe 1 confondue avec 8 ou 9 : Pixels partagés dans les régions centrales.
- Classe 3 avec 5 ou 8 : Formes courbées similaires.
- **Ridge (L2) :** Les coefficients diminuent progressivement vers zéro mais ne l'atteignent jamais exactement. La décroissance est douce et continue.
- Classe 8 avec d'autres : Complexité de la forme avec des trous.
- **Lasso (L1) :** Les coefficients peuvent devenir exactement zéro pour des valeurs élevées de $\lambda$, effectuant une sélection de features.

On observe que certains coefficients restent non-nuls tandis que d'autres sont annulés.

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

## 9.2 Réseau de Neurones (Dataset MNIST)

## 9.3 Régression Logistique avec Scikit-Learn

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

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.

## 9.4 Réseau de Neurones

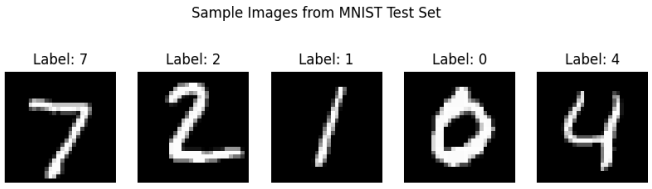



FIGURE 6 – Exemples du dataset MNIST - Images 28x28 pixels
Les erreurs plus nombreuses reflètent la complexité du dataset MNIST (28x28 vs 8x8). Les confusions entre 2/3, 3/5, 4/9, etc., montrent que le modèle simple n'a pas assez de capacité pour distinguer finement. Le manque d'époques et de neurones limite l'apprentissage de features discriminantes.

# 10 Réflexion sur le Travail Effectué

— Classe 0 : 97.14% (952/979 correctes)

## 10.2 Limites du Travail

— Classe 1 : 98.06% (1113/1135)
— Classe 2 : 90.99% (939/1032)
— Classe 3 : 93.17% (941/1010)

— Modèle simple : La régression logistique est limitée aux séparations linéaires ; elle ne capture pas les non-linéarités complexes des chiffres manuscrits.
— Classe 4 : 96.33% (946/982)
— Dataset limité : Digits a seulement 1800 échantillons ; MNIST offre plus de données mais nécessite des modèles plus complexes.
— Classe 5 : 89.13% (795/892)
— Évaluation : Pas de validation croisée systématique ; les résultats dépendent de la séparation train/test.
— Classe 6 : 97.29% (932/958)
— Interprétabilité vs Performance : La régression logistique est interprétable mais moins performante que les réseaux de neurones sur des tâches complexes.
— Classe 7 : 93.87% (965/1028)
— Généralisation : Les modèles sont testés sur des datasets similaires ; la généralisation à d'autres écritures reste à vérifier.
— Classe 8 : 92.20% (898/974)
— Classe 9 : 90.98% (918/1009)

### 10.3 Perspectives d'Amélioration

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

## 11 Conclusion

## 12 Influence des Paramètres

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.

### 12.1 Régression Logistique From Scratch