# Rapport de TP : Régression & Machine Learning

Statistiques

Aymane REGGOUG
Reda HEDDOUN

19 décembre 2025

**Enseignant :** P. Vallet

---

## Table des matières

# Introduction

L'objectif de ce TP est d'étudier la régression linéaire et ses liens avec le machine learning. On va voir comment utiliser des techniques de régression pour faire de la classification binaire et multi-classes, qui sont des problèmes classiques en machine learning

Le rapport traite d'abord des moindres carrés (OLS), puis des régularisations Ridge et Lasso pour stabiliser les modèles quand il y a trop de bruit. Enfin, on teste la régression logistique sur des données synthétiques et sur la base MNIST.

# 1 Régression OLS : Étude de cas

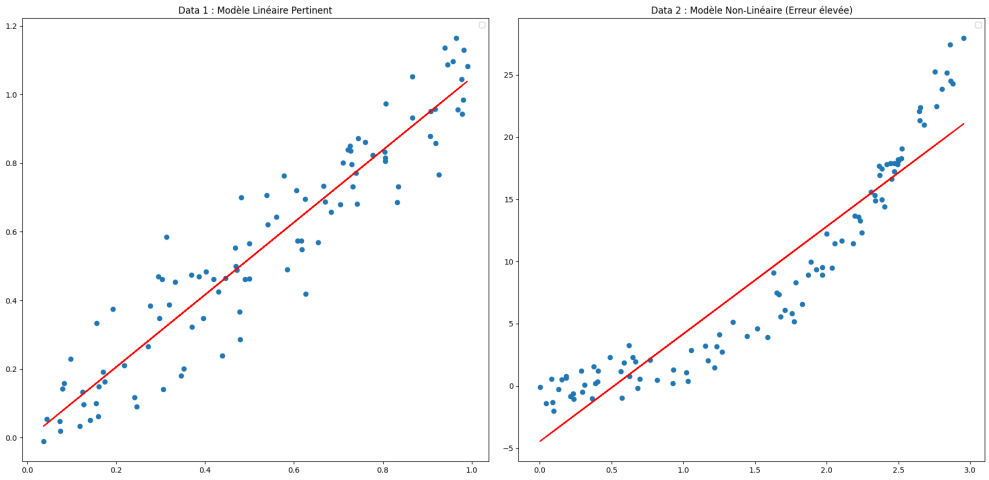


FIGURE 1 – Régression OLS sur data1 (linéaire) et data2 (non-linéaire)

## 1.1 Performance sur données linéaires (data1.npy)

On commence par appliquer OLS avec la perte quadratique $\ell(u, v) = (u - v)^2$ sans régularisation ($r(f) = 0$).

Sur `data1.npy`, les points suivent une droite, donc le modèle linéaire marche bien. L'erreur d'apprentissage est très faible ($R \approx 0.0105$), ce qui confirme qu'un modèle affine suffit ici.

## 1.2 Limites du modèle linéaire (data2.npy)

En refaisant l'expérience sur `data2.npy`, on voit graphiquement que les données suivent une courbe. Le modèle linéaire ne peut pas capturer ça, d'où une erreur beaucoup plus élevée ($R \approx 8.502$).

Ça montre qu'il faut utiliser un espace de redescription polynomial pour transformer les données dans un espace où le modèle linéaire redevient efficace.

### 1.3 Extension à l'espace de redescription (Question 3)

Quand le modèle est non-linéaire, on utilise une fonction $\phi : \mathbb{R}^p \mapsto \mathbb{R}^q$ pour projeter les données dans un espace de dimension plus grande $q > p$. Le modèle devient alors :

$$f(x) = \beta_0 + \sum_{j=1}^{q} \beta_j \phi_j(x) \tag{1}$$

**Comment trouver les coefficients optimaux ?**

On cherche à minimiser le risque empirique $R(\boldsymbol{\beta}) = \frac{1}{n}\|y - \Phi\boldsymbol{\beta}\|_2^2$, où $\Phi$ est la matrice de design définie par :

$$\Phi = \begin{pmatrix} 1 & \phi_1(x_1) & \dots & \phi_q(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(x_n) & \dots & \phi_q(x_n) \end{pmatrix} \in \mathbb{R}^{n \times (q+1)} \tag{2}$$

Comme en TD, on annule le gradient pour obtenir les équations normales $\Phi^T\Phi\boldsymbol{\beta} = \Phi^T y$. La solution est :

$$\hat{\boldsymbol{\beta}} = (\Phi^T\Phi)^{-1}\Phi^T y \tag{3}$$

**Hypothèses nécessaires**

Pour que cette solution existe, il faut que :

— $\Phi^T\Phi$ soit inversible, donc $\Phi$ de rang plein $(q+1)$. En pratique, il faut $n \geq q+1$.

— Les fonctions $\phi_j$ soient linéairement indépendantes. Pour un polynôme ($\phi_j(x) = x^j$), c'est généralement OK si les $x_i$ sont distincts.

*Remarque : Si ces conditions ne sont pas respectées, on peut avoir des problèmes numériques ou plusieurs solutions possibles.*

### 1.4 Résultats et visualisation du surapprentissage (Question 4 et 5)

On teste maintenant un modèle polynomial d'ordre $q = 10$ sur `data2.npy` (peu bruité) et `data3.npy` (très bruité avec peu de points).

**Ce qu'on observe**

En comparant les deux graphes, plusieurs choses sautent aux yeux :

**Sur data2 (gauche)** : Bien que $q = 10$ soit élevé, la courbe reste stable. L'erreur d'apprentissage ($\approx 1.1306$) est cohérente avec ce qu'on avait avant. Le modèle ne dérive pas trop.

**Sur data3 (droite)** : C'est le chaos. La courbe rouge oscille violemment pour passer au plus près de chaque point bruité. L'erreur chute artificiellement à $\approx 0.056$, mais le modèle perd complètement le sens. Aux bords du domaine (vers $x = 0$ et $x = 3$), on voit des oscillations qui partent dans tous les sens.

**Pourquoi ?** Les coefficients $\beta$ explosent pour compenser le bruit. Sans régularisation, minimiser l'erreur quadratique sur des données bruitées avec un modèle complexe donne une solution instable qui ne peut pas généraliser.

### 1.5 Régularisation Ridge : Formulation et Résolution (Question 6 & 7)

Pour éviter l'instabilité vue avant, on introduit Ridge qui pénalise les coefficients trop grands.

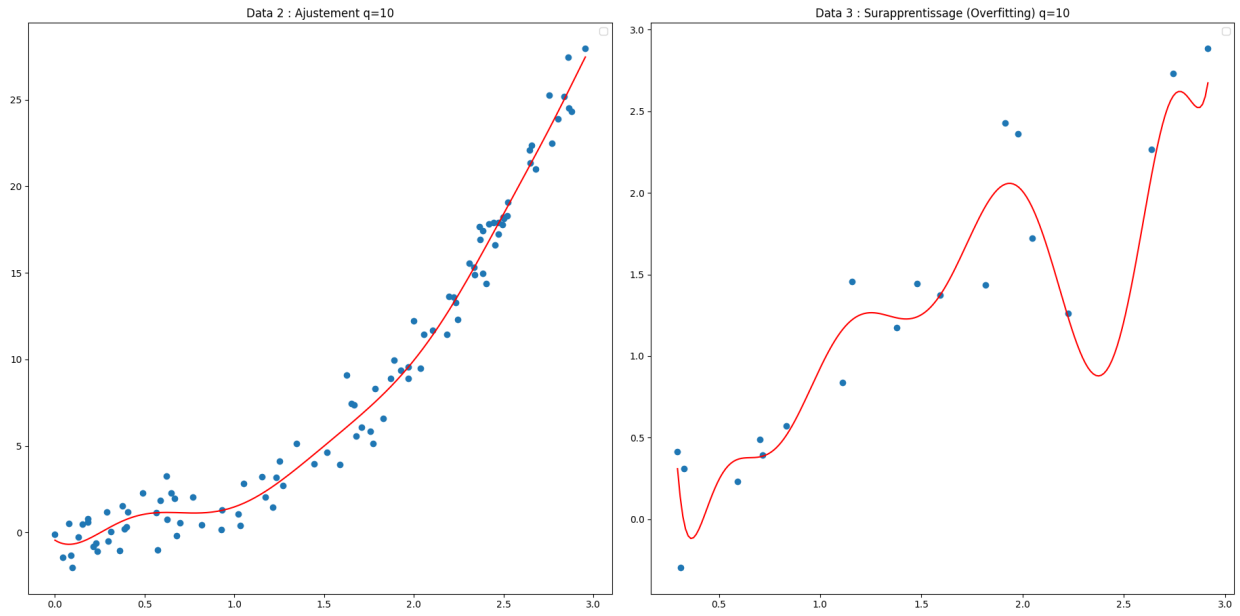


FIGURE 2 – Ajustement polynomial ($q = 10$) sur data2 (gauche) et data3 (droite)

**Le problème à résoudre**

On minimise :

$$J(\beta_0, \beta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta^T x_i)^2 + \lambda\left(\beta_0^2 + \|\beta\|_2^2\right) \tag{4}$$

où $\lambda > 0$ contrôle la force de la régularisation. Plus $\lambda$ est grand, plus on contraint les coefficients.

**Calcul de la solution**

En notant $X$ la matrice de design (avec la colonne de uns) et $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_p)^T$, on réécrit :

$$J(\boldsymbol{\beta}) = \frac{1}{n}\|y - X\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 \tag{5}$$

On dérive par rapport à $\boldsymbol{\beta}$ et on annule :

$$\nabla_{\boldsymbol{\beta}} J(\boldsymbol{\beta}) = \frac{2}{n}X^T(X\boldsymbol{\beta} - y) + 2\lambda\boldsymbol{\beta} = 0 \tag{6}$$

Après simplification, on obtient :

$$(X^T X + n\lambda I)\boldsymbol{\beta} = X^T y \tag{7}$$

D'où la solution Ridge :

$$\hat{\beta}_{Ridge} = (X^T X + n\lambda I)^{-1} X^T y \tag{8}$$

*Point important : Contrairement à OLS, cette solution existe toujours car $X^T X + n\lambda I$ est définie positive (donc inversible) dès que $\lambda > 0$, même si $X^T X$ ne l'est pas. Ça stabilise numériquement le calcul.*

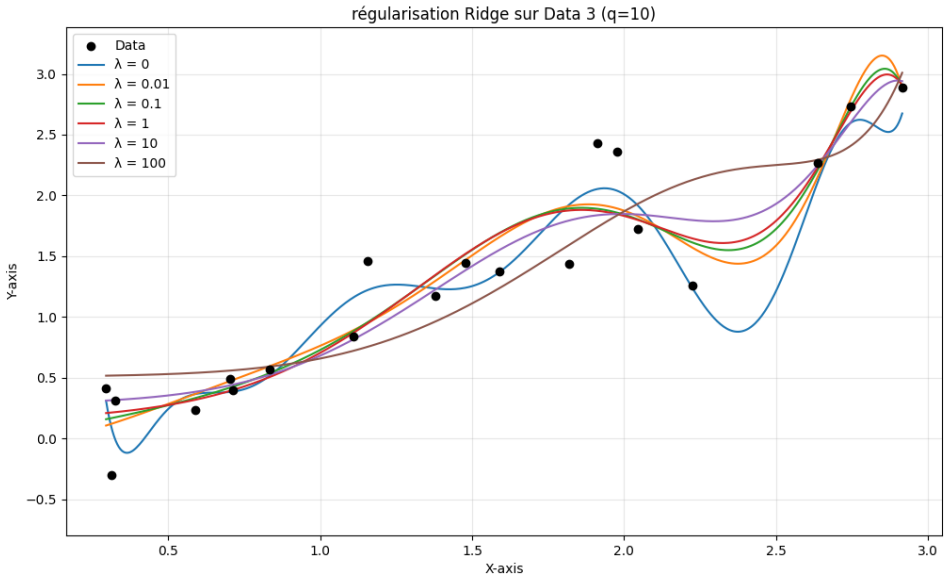


FIGURE 3 – Évolution de l'ajustement Ridge ($q = 10$) sur data3 selon $\lambda$

## 1.6 Influence de $\lambda$ sur Ridge (Question 7)

On teste Ridge sur `data3.npy` avec $q = 10$ en faisant varier $\lambda$ (voir Figure 3).

### Résultats numériques

L'erreur augmente quand $\lambda$ augmente :

| $\lambda$ | 0 (OLS) | 0.01 | 0.1 | 1 | 10 | 100 |
|---|---|---|---|---|---|---|
| **Erreur** | 0.0561 | 0.0756 | 0.0798 | 0.0831 | 0.0929 | 0.1622 |

### Interprétation

$\lambda \to 0$ **(surapprentissage)** : L'erreur est minimale (0.0561) mais la courbe bleue oscille partout. Le modèle mémorise le bruit. C'est un cas de faible biais mais forte variance.

$0.01 \leq \lambda \leq 1$ **(zone de compromis)** : L'erreur augmente un peu, mais les oscillations disparaissent. La courbe se lisse et capture la vraie tendance sans le bruit. C'est ici qu'on trouve le meilleur compromis.

$\lambda \geq 100$ **(sous-apprentissage)** : La pénalité devient trop forte. La courbe s'aplatit excessivement (voir courbe marron) et ne suit plus la dynamique des données. L'erreur remonte à 0.1622 à cause d'un biais trop élevé.

En résumé, Ridge stabilise l'inversion matricielle et améliore la généralisation en "calmant" les coefficients des termes de haut degré.

## 1.7 Régularisation LASSO (Question 8)

Le LASSO utilise une pénalité $\ell_1$ ($\lambda \sum |\beta_j|$) au lieu de $\ell_2$. La différence majeure : LASSO annule certains coefficients, ce qui simplifie le modèle.

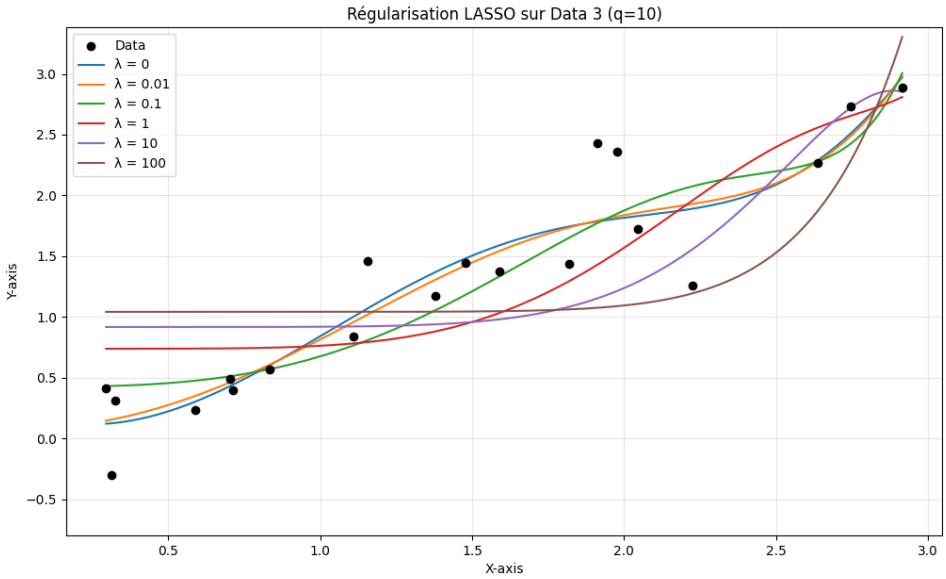


FIGURE 4 – Évolution LASSO ($q = 10$) sur data3 selon $\lambda$

**Résultats et parcimonie**

| $\lambda$ | 0 | 0.01 | 0.1 | 1 | 10 | 100 |
|---|---|---|---|---|---|---|
| **Erreur** | 0.0929 | 0.0947 | 0.1345 | 0.2519 | 0.3649 | 0.4673 |
| **Coeffs nuls** | 0/10 | 3/10 | 4/10 | 6/10 | 8/10 | 9/10 |

*Note : Dès $\lambda = 0.01$, 3 coefficients sont annulés. À $\lambda = 100$, il n'en reste qu'un seul actif !*

**Ce qu'on en retire**

**Sélection automatique** : Contrairement à Ridge qui réduit tous les coefficients, LASSO en met certains à zéro. Ça simplifie le modèle et le rend plus interprétable.

**Stabilité graphique** : Les courbes LASSO (Figure 4) sont plus rigides que Ridge. Pour $\lambda = 1$ (courbe rouge), le modèle a déjà écarté la plupart des puissances de $x$.

**Convergence** : Les warnings pour les petites valeurs de $\lambda$ montrent l'instabilité numérique de la norme $\ell_1$ quand elle n'est pas assez contrainte.

**Bilan** : Là où Ridge "écrase" les coefficients sans les annuler, LASSO fait du tri. On a un modèle plus simple au prix d'une erreur légèrement plus élevée.

## 2 Classification binaire par régression

On évalue maintenant si les méthodes de régression linéaire peuvent résoudre des problèmes de classification en traitant les étiquettes comme des valeurs $y \in \{-1, 1\}$.

## 2.1 Dérivation de la solution (Question 1)

Pour trouver l'hyperplan séparateur, on minimise :

$$J(\boldsymbol{\beta}) = \frac{1}{n}\|y - X\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 \tag{9}$$

où $X \in \mathbb{R}^{n\times(p+1)}$ inclut l'intercept et $y \in \{-1, 1\}^n$.

En développant, on a :

$$J(\boldsymbol{\beta}) = \frac{1}{n}(y^T y - 2\boldsymbol{\beta}^T X^T y + \boldsymbol{\beta}^T X^T X\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta} \tag{10}$$

Le gradient est :

$$\nabla_{\boldsymbol{\beta}} J = \frac{2}{n}(X^T X\boldsymbol{\beta} - X^T y) + 2\lambda\boldsymbol{\beta} \tag{11}$$

En l'annulant, on obtient :

$$\hat{\boldsymbol{\beta}}_{Ridge} = (X^T X + n\lambda I)^{-1} X^T y \tag{12}$$

Pour classer un nouveau point $x$, on calcule $\hat{y} = \text{sign}(\hat{\beta}_0 + \hat{\beta}^T x)$.

## 2.2 Tests de robustesse (Question 2)

On teste OLS ($\lambda = 0$) et Ridge ($\lambda = 500$) sur trois configurations synthétiques avec des outliers croissants.

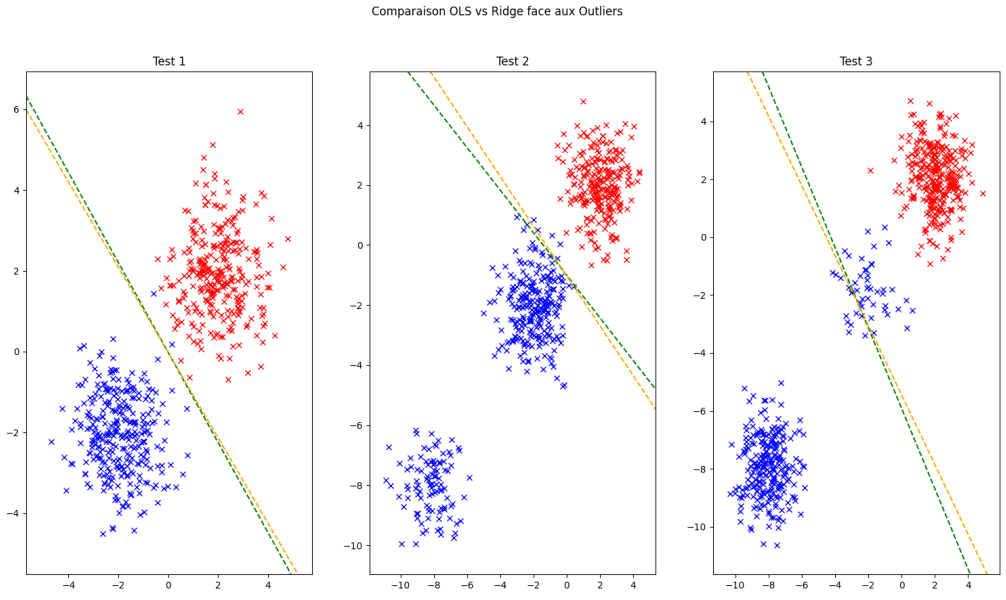


FIGURE 5 – Hyperplans OLS (vert) et Ridge (orange) sur trois configs

**Analyse des résultats**

**Test 1 (Distribution idéale)** : Les classes sont bien séparées. Les deux hyperplans (vert et orange) sont presque confondus et passent bien au milieu.

**Test 2 (Biais de densité)** : Le décalage des points de classe 0 commence à attirer les frontières. Ridge ($\lambda = 500$) subit déjà une rotation visible.

**Test 3 (Effet des Outliers)** : C'est là que ça se gâte. L'introduction d'une masse de points aberrants en $(-8, -8)$ fait pivoter violemment les deux hyperplans (vert OLS et orange Ridge) vers le bas.

**Pourquoi ça échoue ?**

La perte quadratique $(y_i - f(x_i))^2$ pénalise l'écart au carré. Pour des outliers très éloignés, l'erreur devient énorme et domine le coût total. Le modèle essaie de réduire cette erreur en s'orientant vers les outliers, même avec Ridge ($\lambda = 500$).

## 2.3 Régression Logistique (Question 3)

La régression logistique utilise la perte $\ell(m) = \log(1 + e^{-m})$ au lieu de la perte quadratique.

**Pourquoi c'est mieux ?**

Sur la Figure 6, l'ajout de la régression logistique (ligne violette) montre un comportement complètement différent :

**Résistance à la rotation** : Alors que OLS (vert) et Ridge (orange) pivotent vers les outliers en $(-8, -8)$ dans le Test 3, la frontière logistique reste stable et bien positionnée entre les deux classes principales.

**Saturation du gradient** : La perte logistique a un gradient qui sature (tend vers zéro) pour les points bien classés et éloignés. Une fois qu'un outlier est identifié avec certitude, son influence devient négligeable. Le modèle l'ignore.

## 2.4 Performances comparées (Question 4)

Sur les Tests 1 et 2 (distributions Gaussiennes propres), les trois modèles se comportent pareil. Les frontières sont presque superposées, passant au centre des nuages.

## 2.5 Impact de l'augmentation des outliers (Question 5)

Le Test 3 montre clairement la supériorité de la logistique quand $n_{out}$ augmente :

**Échec d'OLS et Ridge** : Les hyperplans basés sur la perte quadratique subissent une rotation marquée car ils donnent trop d'importance aux points loin de l'hyperplan.

**Robustesse de la Logistique** : La ligne violette reste quasi immobile malgré les outliers. Comme ils sont loin de la zone de décision, la perte logistique sature et ignore leur influence. C'est clairement mieux adapté pour la classification sur des données bruitées.

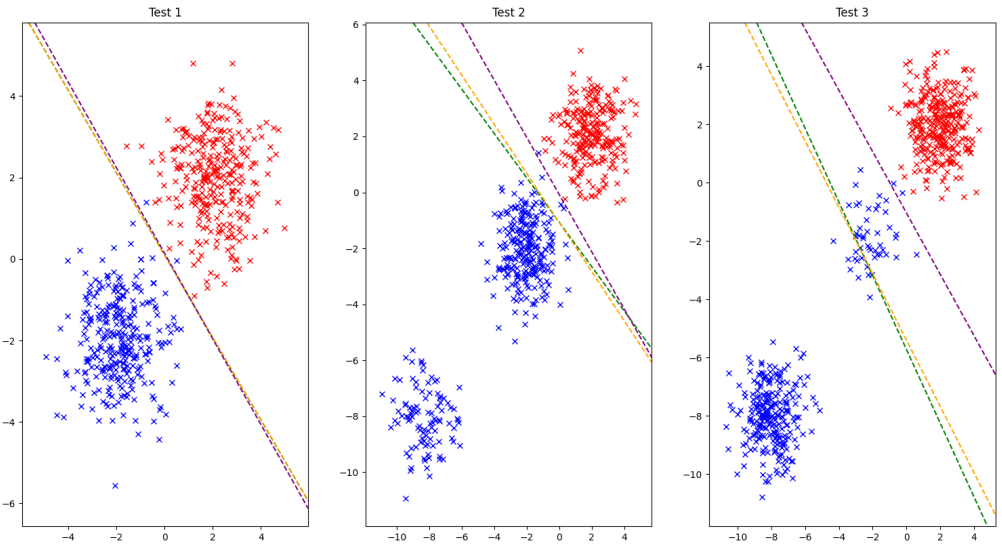


FIGURE 6 – OLS (vert), Ridge (orange, $\lambda = 500$) et Logistique (violet) face aux Outliers

# 3 Tests sur la base de données MNIST

## 3.1 Présentation de l'expérimentation MNIST

L'objectif de cette partie est d'évaluer les performances de la régression logistique sur un problème de classification multi-classes réel et complexe : la base de données MNIST. Cette base se compose de 70 000 images de chiffres manuscrits (de 0 à 9), chaque image étant un vecteur de 784 pixels. Pour réussir cette tâche, le protocole suivant a été appliqué :

— **Normalisation :** Les données sont centrées et réduites à l'aide de `StandardScaler` afin que chaque pixel contribue de manière équitable à l'apprentissage.

— **Apprentissage :** Le modèle minimise une fonction de perte logistique $l(m) = log(1 + e^{-m})$ associée à une régularisation $l^2$ pour stabiliser les coefficients $\beta$.

— **Évaluation :** Une fois le modèle entraîné sur l'ensemble `X_train`, nous testons sa capacité de généralisation sur l'ensemble `X_test` en utilisant une matrice de confusion.

## 3.2 Matrice de confusion

L'analyse de la matrice de confusion obtenue permet d'évaluer précisément les performances du classifieur logistique multiclasse sur la base MNIST :

— **Performances Globales :** On observe une forte concentration des valeurs sur la diagonale principale, ce qui indique que la régression logistique est très efficace pour ce problème de classification d'images.

— **Chiffres les mieux reconnus :**Le chiffre 1 présente le score de reconnaissance le plus élevé (1923 bonnes prédictions), suivi du 0 (1633). Ces chiffres possèdent des structures géométriques simples et très distinctes des autres classes.

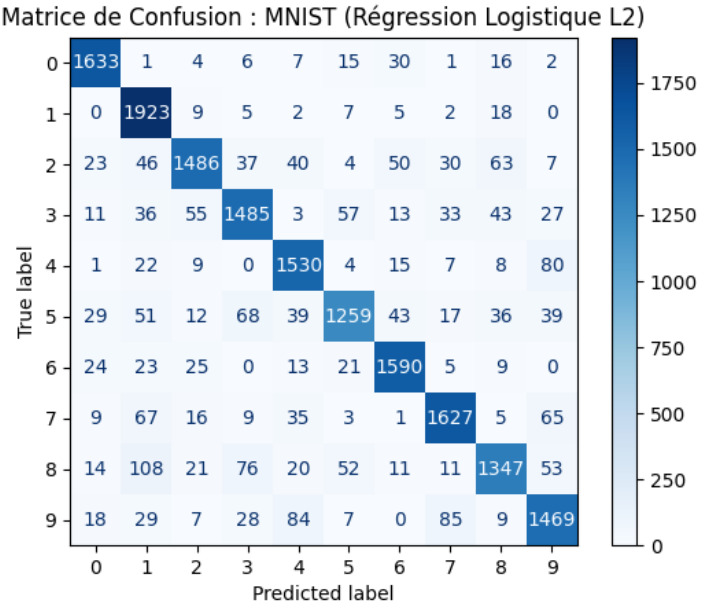


FIGURE 7 – Matrice de confusion pour les données MNIST

— **Confusions :**Les erreurs de classification (valeurs hors-diagonale) ne sont pas aléatoires mais liées à des similarités de forme entre certains chiffres :

  — **4 vs 9 :** On note une confusion notable entre le 4 et le 9 (80 erreurs). Cela s'explique par la présence commune d'une boucle supérieure et d'un trait vertical (*cf.* Fig8).

  — **8 vs 1 :** Le chiffre 8 est parfois confondu avec le 1 (108 erreurs).

  — **7 vs 9 :** Le 7 est régulièrement pris pour un 9 (65 erreurs).

### 3.3 Visualisation des coefficients $\hat{\beta}$

### Analyse de la visualisation des coefficients $\hat{\beta}$

#### 1. Visualisation avec Régularisation $L^2$

L'examen des images des coefficients montre que les poids sont répartis sur l'ensemble de la forme du chiffre, offrant une vision "lissée" de la structure :

— **Zones Rouges :** Pixels ayant des poids positifs favorisant l'appartenance à la classe. On y distingue nettement le « squelette » moyen de chaque chiffre.

— **Zones Bleues :** Pixels ayant des poids négatifs défavorisant la classe. Ils se situent généralement aux endroits où d'autres chiffres possèdent des traits caractéristiques alors que le chiffre concerné n'en possède pas.

#### 2. Visualisation avec Régularisation $L^1$ (LASSO)

La comparaison avec les résultats de la régularisation $L^1$ met en évidence des propriétés mathématiques différentes :

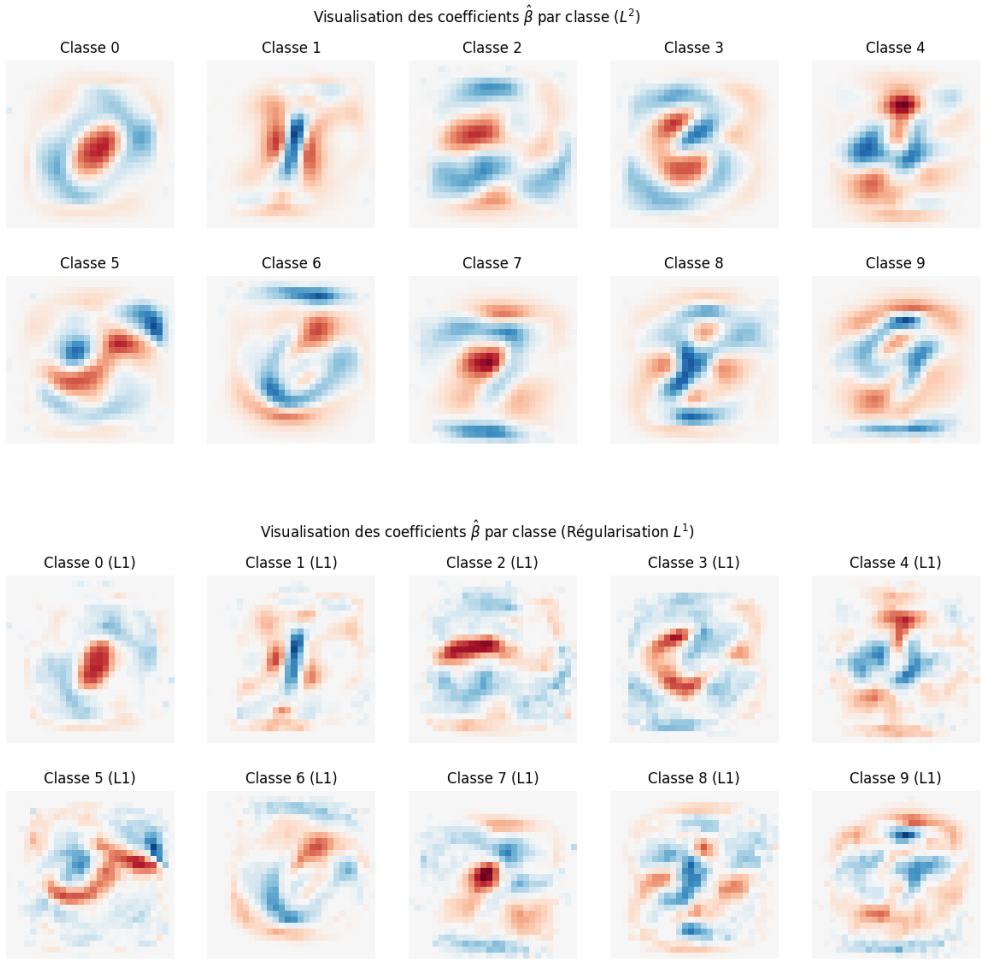


FIGURE 8 – Les coefficients $\hat{\beta}$ dans les 2 cas de régularisation de la regression logistique

— **Parcimonie :** Les images $L^1$ apparaissent plus « bruitées » ou « pointillées » par des pixels blancs. Cela s'explique par la nature de la norme $L^1$ qui force de nombreux coefficients $\hat{\beta}_i$ à être exactement nuls.

— **Sélection de variables :** Le modèle $L^1$ opère une sélection automatique des pixels. Il ne conserve que les points les plus discriminants (points stratégiques) pour identifier le chiffre, supprimant ainsi les informations spatiales redondantes ou moins informatives.

### 3.4 Conclusion : Classification sur la base MNIST

L'étude de la base MNIST par régression logistique met en évidence trois points majeurs :

— **Efficacité de la perte logistique :** En optimisant la marge $m(x, y) = y(\beta_0 + \beta^T x)$, ce modèle s'avère bien plus robuste qu'une approche par moindres carrés pour la classification. Les résultats de la matrice de confusion montrent une excellente reconnaissance globale malgré quelques **confusions** inhérentes à la proximité de certains chiffres (ex : 4 vs 9).

— **Impact des régularisations :**

— La norme $\ell^2$ stabilise l'apprentissage en répartissant les poids sur tout le « squelette » du chiffre.
— La norme $\ell^1$ **(LASSO)** apporte de la **parcimonie**, ne conservant que les pixels les plus discriminants.

En conclusion, la régression logistique, bien que linéaire dans son espace de décision, constitue un classifieur robuste et hautement interprétable, capable d'allier précision mathématique et interprétabilité visuelle des coefficients $\hat{\beta}$.

## Conclusion

Ce TP nous a permis de faire le lien entre les méthodes statistiques classiques vues en cours et leurs applications concrètes en machine learning.

L'aspect le plus intéressant a été de voir comment des modèles simples peuvent être adaptés à des problèmes complexes en jouant sur la régularisation et le choix de la fonction de perte. Les expérimentations sur données synthétiques et sur MNIST ont montré qu'il existe toujours un compromis à trouver entre la capacité du modèle à s'adapter aux données et sa stabilité face au bruit.

Au final, ce TP nous a donné une bonne base pour comprendre les choix à faire quand on construit un modèle d'apprentissage : quand privilégier la simplicité, comment gérer le surapprentissage, et pourquoi l'interprétabilité reste importante même face à des méthodes plus performantes.