Université Ibn Zohr
Faculté pluridisciplinaire de Ouarzazate
Filière master : Intelligence artificielle et applications

Master IAA – 2025

TP régression et classification par la descente de gradient

Projet tutoré présenté par :

HANANE IQLI
GHIZLANE AIT HADDOU

Sous la direction de :

Prof. **AI.HADRI**

Année universitaire : 2024-2025

Table des matières

Introduction **4**

1 La régression linéaire et non linéaire **5**
1.1 La régression linéaire . . . 5
1.1.1 Modélisation et formulation mathématique . . . 5
1.1.2 Méthodes de descente de gradient . . . 6
1.1.3 prétraitement du dataset housing . . . 6
1.2 Implémentation de la régression linéaire à pas fixe . . . 7
1.2.1 Initialisation des Paramètres . . . 7
1.2.2 Configuration de l'optimisation . . . 8
1.3 Implémentation de la régression linéaire à pas optimal . . . 9
1.3.1 Méthode de recherche linéaire pour choisir le pas optimal . . . 9
1.3.2 Résultats de la descente de gradient à pas optimal . . . 9
1.4 La régression non linéaire . . . 10
1.4.1 Modélisation et formulation mathématique . . . 10
1.4.2 Méthodes de descente de gradient . . . 11
1.5 Implémentation de la régression non linéaire à pas fixe, pas optimal. . . . 11
1.5.1 Initialisation des paramètres . . . 11
1.5.2 Résultats de la descente de gradient non linéaire à pas fixe . . . 12
1.5.3 Résultats de la descente de gradient non linéaire à pas optimal . . . 13

2 Classification avec réseau de Neurones sur MNIST **15**
2.1 Classification avec réseau à une couche . . . 15
2.1.1 Chargement et préparation des Données MNIST . . . 16
2.1.2 Normalisation : $X/255$ . . . 16
2.1.3 Encodage One-Hot des étiquettes . . . 17
2.2 Implémentez le réseau à pas fixe. . . . 17
2.2.1 Initialisation et paramètres d'entraînement . . . 17
2.3 Implémentez le réseau à pas optimal. . . . 18
2.3.1 Initialisation et paramètres d'entraînement . . . 18
2.3.2 Initialisation et paramètres d'entraînement . . . 19
2.4 Remarque . . . 19
2.5 Classification avec Réseau Multicouche . . . 20
2.5.1 Résultats de gradient descent à pas fixe . . . 20
2.6 Remarque . . . 21

2.6.1 Résultats de gradient descent à pas optimal 21
2.6.2 Initialisation des paramètres 21
2.6.3 Initialisation des paramètres 23
2.7 Remarque 23

Introduction

L'apprentissage automatique repose sur des méthodes mathématiques et algorithmiques permettant à un modèle d'apprendre à partir de données pour effectuer des prédictions ou des classifications. Parmi les approches fondamentales en apprentissage supervisé, la régression et la classification occupent une place centrale. Ces deux problèmes peuvent être résolus à l'aide d'algorithmes d'optimisation, dont la descente de gradient, une technique clé pour minimiser une fonction de coût et ajuster les paramètres d'un modèle.
Ce TP a pour but d'implémenter et d'expérimenter avec la descente de gradient appliquée à deux problèmes majeurs :
la régression linéaire et non linéaire.
La classification avec réseau à une couche et d'autre avec réseau multicouche.
L'objectif est de comprendre comment fonctionne l'optimisation par descente de gradient, comment choisir les hyperparamètres (comme le taux d'apprentissage) et comment évaluer les performances des modèles.

Chapitre 1

La régression linéaire et non linéaire

Introduction

Ce TP a pour objectif d'étudier et de comparer deux approches de régression — linéaire et polynomiale — appliquées au dataset California Housing Prices, en utilisant la méthode de descente de gradient. La régression linéaire permettra de modéliser des relations simples entre les caractéristiques des logements (comme le revenu médian ou l'âge du quartier) et leur prix médian, tandis que la régression polynomiale (de degré 2 ou 3) sera employée pour capturer des relations non linéaires plus complexes. Nous implémenterons deux versions de l'algorithme de descente de gradient : l'une avec un pas d'apprentissage fixe (a =0,1 ou 0.001) et l'autre avec un pas optimal déterminé par recherche linéaire, afin d'analyser leur impact sur la vitesse de convergence et la précision des prédictions. Les performances des modèles seront évaluées à l'aide de métriques telles que le RMSE, le MAE et le coefficient R^2, et les résultats seront visualisés pour comparer leur capacité à ajuster les données. Ce travail mettra en lumière l'importance du choix des hyperparamètres et des techniques d'optimisation dans la construction de modèles prédictifs efficaces, tout en fournissant des pistes pour améliorer leur robustesse et leur interprétabilité.

1.1 La régression linéaire

1.1.1 Modélisation et formulation mathématique

La régression linéaire a pour objectif d'établir une relation linéaire entre les variables explicatives $\mathbf{X}$ (telles que `median_income`, `housing_median_age`) et la variable cible y (`median_house_value`). Le modèle s'exprime sous la forme :

$$y = \mathbf{W}\mathbf{X} + B \tag{1.1}$$

où :

— $\mathbf{W}$ est la matrice des poids (coefficients).

— B représente le terme de biais (constant).

— $\mathbf{X}$ désigne la matrice des caractéristiques normalisées.

Fonction de coût (MSE)

La fonction de coût quadratique est donnée par :

$$J(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{1.2}$$

Gradients

Les gradients de la fonction de coût par rapport aux paramètres sont :

$$\nabla_{\mathbf{w}} J = \frac{2}{n}\mathbf{X}^T(\hat{\mathbf{y}} - \mathbf{y}) \tag{1.3}$$

$$\frac{\partial J}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i) \tag{1.4}$$

1.1.2 Méthodes de descente de gradient

Descente à Pas Fixe

La mise à jour des paramètres avec un taux d'apprentissage constant α s'effectue comme suit :

$$\begin{aligned} \mathbf{w} &\leftarrow \mathbf{w} - \alpha\frac{\partial J}{\partial \mathbf{w}} \\ b &\leftarrow b - \alpha\frac{\partial J}{\partial b} \end{aligned} \tag{1.5}$$

Descente à Pas Optimal (Recherche Linéaire)

Le pas optimal α_k à chaque itération est déterminé en minimisant la fonction de coût :

$$\alpha_k = \arg\min_{\alpha} J\left(\mathbf{w} - \alpha\frac{\partial J}{\partial \mathbf{w}}, b - \alpha\frac{\partial J}{\partial b}\right) \tag{1.6}$$

où :

— α_k est choisi parmi un ensemble de valeurs candidates (par exemple $\{0.001, 0.01, 0.1\}$)

— La minimisation se fait par évaluation directe J pour différentes valeurs de α

1.1.3 prétraitement du dataset housing

Nous avons choisi quatre colonnes, trois ['median-income', 'total-rooms', 'housing-median-age']. comme des variables et pour la variable cible est ['median-house-value'].

	median_income	total_rooms	housing_median_age	median_house_value
0	8.3252	880.0	41.0	452600.0
1	8.3014	7099.0	21.0	358500.0
2	7.2574	1467.0	52.0	352100.0
3	5.6431	1274.0	52.0	341300.0
4	3.8462	1627.0	52.0	342200.0

FIGURE 1.1 – Les cinq premières variables

Normalisation des Données

Pour garantir la stabilité numérique des calculs et permettre une convergence plus rapide de l'algorithme d'optimisation, nous avons normalisé les données :
Normalisation de X (caractéristiques) et y (prix médian des logements)

	median_income	total_rooms	housing_median_age	median_house_value
0	0.539668	0.022331	0.784314	0.902266
1	0.538027	0.180503	0.392157	0.708247
2	0.466028	0.037260	1.000000	0.695051
3	0.354699	0.032352	1.000000	0.672783
4	0.230776	0.041330	1.000000	0.674638

FIGURE 1.2 – data après la normalisation

Division des Données

Les données ont été séparées en deux ensembles distincts :
Ensemble d'entraînement (80 % des données) : pour l'apprentissage du modèle
Ensemble de test (20% des données) : Pour évaluer la performance sur des données non vues
La séparation a été effectuée de manière aléatoire avec une graine fixe (random-state=42) pour assurer la reproductibilité des résultats.

1.2 Implémentation de la régression linéaire à pas fixe

1.2.1 Initialisation des Paramètres

Pour démarrer l'algorithme de descente de gradient, nous avons initialisé les paramètres du modèle de la manière suivante :

— **Poids** ($\mathbf{w}$) : vecteur de taille 3 (correspondant aux 3 caractéristiques sélectionnées) initialisé à $\mathbf{0}$

$$\mathbf{w} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

— **Biais** (b) : Scalaire initialisé à 0

$$b = 0$$

Cette initialisation simple garantit un point de départ neutre pour l'optimisation.

1.2.2 Configuration de l'optimisation

L'algorithme a été configuré avec les hyperparamètres suivants :

Paramètre	**Valeur**	**Justification**
Taux d'apprentissage (α)	0.01	Pas assez petit pour éviter la divergence, tout en permettant une convergence raisonnable.
Nombre d'itérations	1000	Suffisant pour observer la convergence du coût dans la plupart des cas.

TABLE 1.1 – Configuration des hyperparamètres

Résultats de la descente de gradient à pas fixe

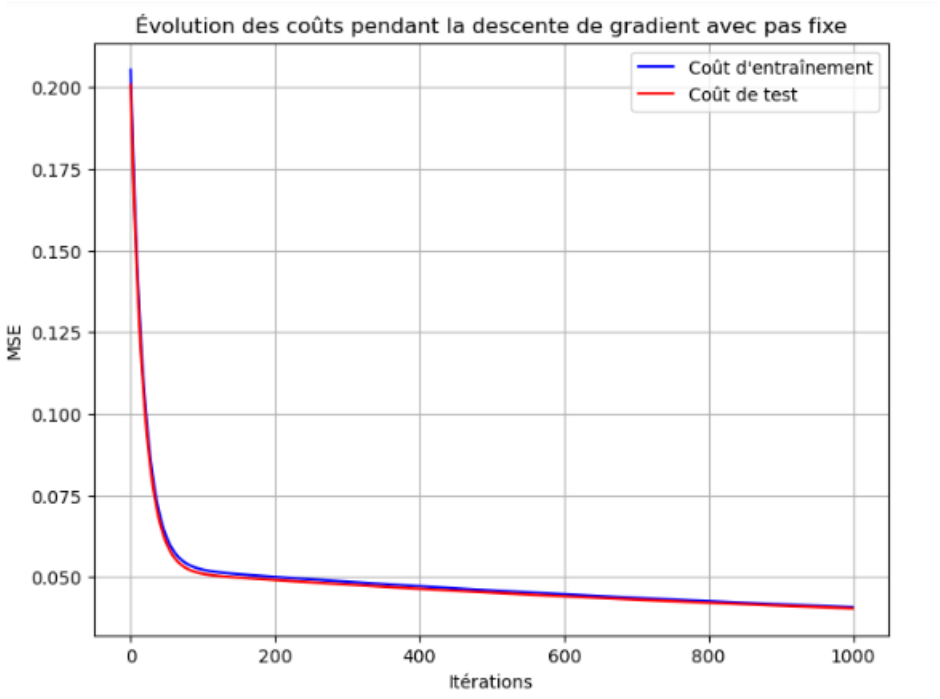


FIGURE 1.3 – évolution de la loss de la descente de gradient à pas fixe

Le graphique illustre l'évolution des coûts d'entraînement et de test lors d'une descente de gradient avec un pas fixe. Les deux courbes diminuent rapidement dans les premières itérations (0–200), puis se stabilisent progressivement à partir de 400 itérations, atteignant des valeurs autour de 0.05–0.075. Cette convergence rapide et stable suggère que le taux d'apprentissage est bien choisi.

les métriques d'évaluation

Métrique	Valeur
RMSE	0.201
MAE	0.1593
R^2	0.2711

TABLE 1.2 – Métriques d'évaluation avec pas fixe

Les métriques d'évaluation indiquent que le modèle présente une performance acceptable. Le MAE (0.1593), inférieur au RMSE ce qui montre que la plupart des erreurs sont concentrées autour de cette valeur, le R^2 (coefficient de détermination) de 0.2711 révèle que seulement 27,11 % de la variabilité des données est expliqué par le modèle, ce qui est relativement faible. d'après les résultats obtenus on remarque que le modèle est trop simple ce qui signifie le manque des features utilisés.

1.3 Implémentation de la régression linéaire à pas optimal

1.3.1 Méthode de recherche linéaire pour choisir le pas optimal

La recherche linéaire (line search) est une technique d'optimisation permettant de déterminer un pas optimal (alpha) à chaque itération de la descente de gradient, afin d'accélérer la convergence tout en garantissant la stabilité. Contrairement à un pas fixe, elle adapte dynamiquement (alpha) pour minimiser la fonction de coût de manière efficace.

1.3.2 Résultats de la descente de gradient à pas optimal

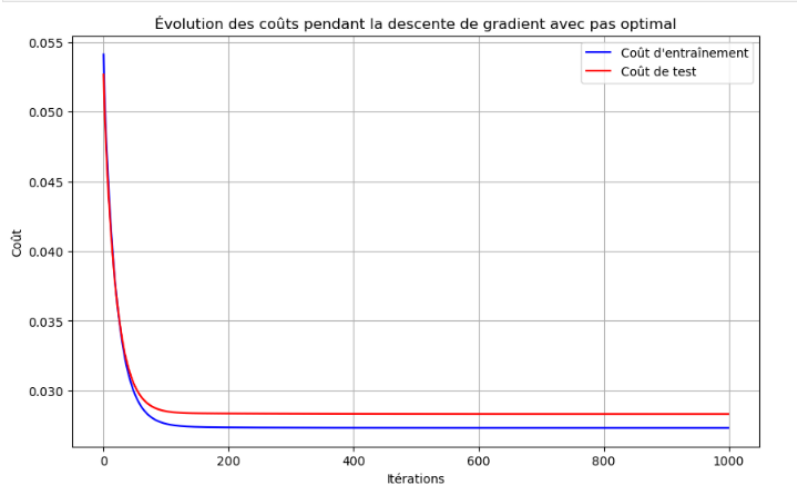


FIGURE 1.4 – Évolution de la loss de la descente de gradient à pas optimal

La courbe montre l'évolution des coûts d'entraînement et de test lors d'une descente de gradient utilisant un pas optimal (probablement déterminé par une méthode de recherche linéaire). Les deux courbes convergent rapidement vers des valeurs basses, avec le coût d'entraînement atteignant environ 0.030 et le coût de test se stabilisant légèrement au-dessus (vers 0.035–0.040) après 1000 itérations. Cette faible différence entre entraînement et test suggère une bonne généralisation du modèle, sans surapprentissage marqué. La décroissance régulière et l'absence d'oscillations indiquent que le pas adaptatif a efficacement équilibré vitesse de convergence et stabilité. Comparé à un pas fixe (où les coûts se stabilisaient autour de 0.05–0.075), l'utilisation d'un pas optimal améliore significativement la performance finale, tout en maintenant un écart contrôlé entre les données d'entraînement et de test. Cela confirme l'intérêt des méthodes de recherche linéaire pour optimiser le taux d'apprentissage.

les métriques d'évaluation

Métrique	**Valeur**
RMSE	0.1683
MAE	0.1253
R^2	0.4891

TABLE 1.3 – Métriques d'évaluation avec pas optimal

Les métriques RMSE (0.1683) et MAE (0.1253) démontrent une amélioration de la précision des prédictions grâce à l'utilisation d'un pas optimal. Le RMSE indique que les erreurs moyennes sont réduites à environ 0.17 unité, avec une diminution significative des erreurs importantes par rapport au pas fixe. Le MAE, quant à lui, confirme que la majorité des erreurs se situe autour de 0.13, révélant une distribution homogène et l'absence d'écarts extrêmes. Par ailleurs, le R^2 de 0.4891 (48.91 %) souligne une meilleure capacité du modèle à expliquer la variance des données, avec près de 49 % de la variabilité capturée, contre seulement 27 % précédemment.

comparaison

L'utilisation d'un pas optimal a drastiquement amélioré la performance du modèle, tant en précision (RMSE/MAE) qu'en capacité explicative (R^2). Ces résultats valident l'intérêt des méthodes de recherche linéaire pour l'optimisation du taux d'apprentissage. Par contre, le pas fixe ne donne pas des résultats précis,La convergence est lente, car les mises à jour des paramètres sont minimales.

1.4 La régression non linéaire

1.4.1 Modélisation et formulation mathématique

— Modèle : Polynôme de degré 3 :

$$\hat{y} = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

— Coût :

$$J(w) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

— Gradient :

$$\frac{\partial J}{\partial w_j} = \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)x_i^j, \quad j = 0, 1, 2, 3$$

1.4.2 Méthodes de descente de gradient

descente de gradient à pas fixe

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha\frac{\partial J}{\partial \mathbf{w}} \tag{1.7}$$

descente de gradient à pas optimal

Le pas optimal α_k à chaque itération est déterminé en minimisant la fonction de coût :

$$\alpha_k = \arg\min_{\alpha} J(\mathbf{w} - \alpha\frac{\partial J}{\partial \mathbf{w}}) \tag{1.8}$$

où :

— α_k est choisi parmi un ensemble de valeurs candidates (par exemple $\{0.001, 0.01, 0.1\}$)

— La minimisation se fait par évaluation directe J pour différentes valeurs de α

1.5 Implémentation de la régression non linéaire à pas fixe, pas optimal.

1.5.1 Initialisation des paramètres

Pour démarrer l'algorithme de descente de gradient, nous avons initialisé les paramètres du modèle de la manière suivante :

— **Poids** ($\mathbf{w}$) : vecteur de taille 3 (correspondant aux 3 caractéristiques sélectionnées) initialisé à $\mathbf{0}$

$$\mathbf{w} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

— **learning rate** =0.01

— **nombre d'itérations** =1000

1.5.2 Résultats de la descente de gradient non linéaire à pas fixe

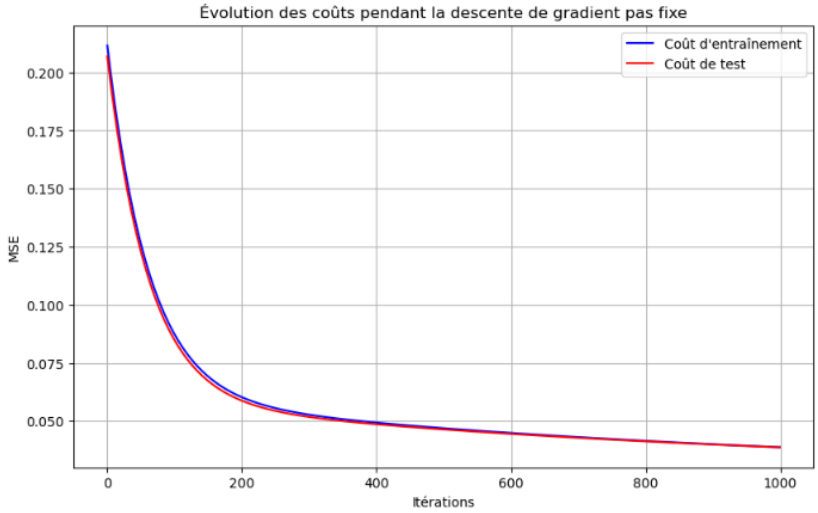


FIGURE 1.5 – évolution de couts

Le graphique montre l'évolution des coûts d'entraînement et de test lors d'une descente de gradient utilisant un pas fixe. Les deux courbes convergent progressivement vers des valeurs faibles, avec le coût d'entraînement atteignant environ 0.050 et le coût de test se stabilisant légèrement au-dessus (vers 0.075-0.100) après 1000 itérations.

les métriques d'évaluations

Métrique	**Valeur**
RMSE	0.1968
MAE	0.1519
R^2	0.3017

TABLE 1.4 – Métriques d'évaluation avec pas fixe

Le modèle de régression non linéaire, entraîné par descente de gradient à pas fixe, converge de manière stable avec une bonne généralisation entre l'entraînement et le test. Les erreurs RMSE (0.1968) et MAE (0.1519) indiquent des prédictions raisonnablement précises, tandis que le R^2 de 0.3017 montre que le modèle explique environ 30% de la variance, laissant une marge d'amélioration possible.

1.5.3 Résultats de la descente de gradient non linéaire à pas optimal

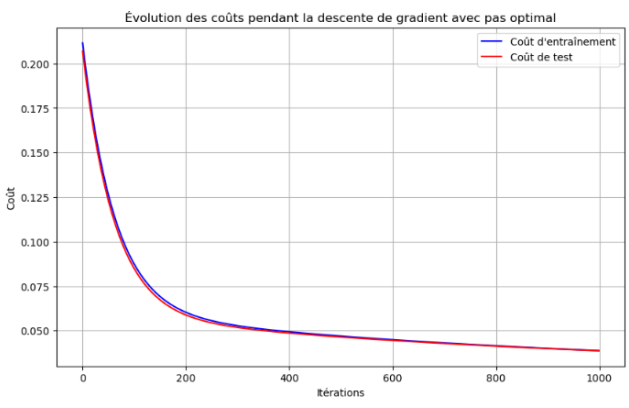


FIGURE 1.6 – Enter Caption

Ce graphique montre l'évolution du coût lors de l'entraînement d'un modèle de régression non linéaire en utilisant une descente de gradient avec pas optimal. On observe une diminution rapide et régulière du coût sur les ensembles d'entraînement (courbe bleue) et de test (courbe rouge), ce qui reflète une convergence efficace. La proximité constante entre les deux courbes indique une très bonne généralisation du modèle.

les métriques d'évaluation

Métrique	**Valeur**
RMSE	0.1968
MAE	0.1519
R^2	0.3017

TABLE 1.5 – Métriques d'évaluation avec pas optimal

Le modèle de régression non linéaire, entraîné par descente de gradient à pas optimal, converge de manière stable avec une bonne généralisation entre l'entraînement et le test. Les erreurs RMSE (0.1968) et MAE (0.1519) indiquent des prédictions raisonnablement précises, tandis que le R^2 de 0.3017 montre que le modèle explique environ 30% de la variance, laissant une marge d'amélioration possible.
On a trouvé les mêmes résultats que le gradient à pas fixe, donc cela signifie que le pas fixe choisi était déjà bien ajusté, proche de l'optimal.

TABLE 1.6 – Résultats de régression linéaire par méthode d'optimisation

Méthode	Métrique	Régression linéaire	Régression non linéaire
Descente de gradient à pas fixe	RMSE	0.2010	0.1968
	MAE	0.1593	0.1519
	R^2	0.2711	0.3017
Descente de gradient à pas optimal	RMSE	0.1683	0.1968
	MAE	0.1253	0.1519
	R^2	0.4891	0.3017

Conclusion

Le choix entre pas fixe et optimal dépend fortement de la complexité du modèle et des données : Le pas fixe reste utile pour des problèmes convexes et bien conditionnés, mais montre vite ses limites. Le pas optimal (ou adaptatif) s'impose dès que la fonction de coût est irrégulière ou multi-modale, garantissant rapidité et robustesse. Cette analyse souligne l'importance de l'optimisation du taux d'apprentissage comme levier clé pour équilibrer précision, vitesse et stabilité dans tout projet de modélisation prédictive.

Chapitre 2

Classification avec réseau de Neurones sur MNIST

Introduction

La classification des chiffres manuscrits à l'aide de réseaux de neurones est un problème classique en apprentissage automatique, souvent abordé avec le jeu de données MNIST. Ce dernier,contient des images en noir et blanc de chiffres de 0 à 9, chacune de taille 28 × 28 pixels.Un réseau de neurones artificiels peut apprendre automatiquement des motifs discriminants à partir des pixels bruts. Dans cette implémentation, nous utilisons un modèle linéaire (équivalent à une régression softmax) entraîné par descente de gradient, où les poids sont optimisés pour minimiser une fonction de coût d'entropie croisée. L'algorithme alterne entre une phase de prédiction (forward pass) et une rétropropagation des erreurs pour ajuster les paramètres. Les performances sont évaluées sur un ensemble de test à l'aide de métriques telles que l'accuracy, la matrice de confusion et le rapport de classification, démontrant ainsi l'efficacité du modèle à généraliser.

2.1 Classification avec réseau à une couche

Cette partie se concentre sur un réseau de neurones à une couche appliqué au jeu de données MNIST. Vous implémenterez la descente de gradient à pas fixe et étudierez l'impact des hyperparamètres (pas d'apprentissage, taille des mini-batches). Le modèle mathématique est défini comme suit :

Modèle

Pour une matrice d'entrée $X \in \mathbb{R}^{n \times 784}$ (où n est le nombre d'exemples et 784 correspond aux dimensions des images 28x28), la sortie prédite est donnée par :

$$\hat{Y} = \text{softmax}(XW + B)$$

où la fonction softmax est définie composante par composante comme :

$$\text{softmax}(z)_{i,j} = \frac{e^{z_{i,j}}}{\sum_{k=1}^{K} e^{z_{i,k}}}$$

Fonction de coût

La fonction de coût utilisée est l'entropie croisée avec régularisation numérique :

$$J = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{10} Y_{i,j}\log\left(\hat{Y}_{i,j} + 10^{-15}\right)$$

Algorithme d'optimisation

— **Descente de gradient à pas fixe** : Mise à jour des paramètres via les gradients calculés.

— **Recherche de pas optimal** : Minimisation de J avec recherche linéaire sur le taux d'apprentissage α.

2.1.1 Chargement et préparation des Données MNIST

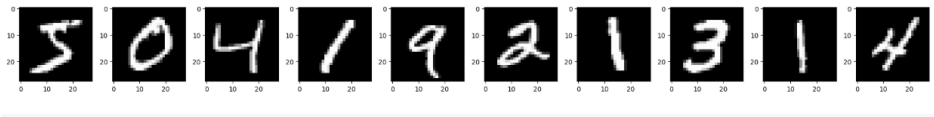


FIGURE 2.1 – visualisation des données

Le dataset MNIST est une base de données de référence en apprentissage automatique. Il contient :

— **60 000 images** pour l'entraînement

— **10 000 images** pour les tests

Chaque image représente un chiffre manuscrit allant de 0 à 9 avec les caractéristiques suivantes :

— Format en **niveaux de gris** (1 seul canal)

— Taille fixe de **28×28 pixels**

2.1.2 Normalisation : $X/255$

Les images MNIST possèdent les caractéristiques suivantes :

— Valeurs initiales des pixels comprises entre :

  — 0 (noir pur)

  — 255 (blanc pur)

Pour optimiser l'apprentissage du réseau de neurones, nous appliquons une normalisation des données :

$$X_{\text{normalisé}} = \frac{X}{255} \tag{2.1}$$

Cette opération a pour effets :

— Transformation des valeurs en nombres flottants

— Intervalle normalisé entre 0 et 1

— Stabilisation de la descente de gradient

— Réduction des problèmes numériques

2.1.3 Encodage One-Hot des étiquettes

Les étiquettes y sont initialement représentées par des entiers (par exemple, $y = 3$ pour désigner le chiffre « 3 »). Dans le cadre d'une classification multiclasse, ces entiers sont convertis en vecteurs *one-hot*. Cela signifie que chaque étiquette est transformée en un vecteur de la taille du nombre total de classes, avec une seule composante égale à 1 (correspondant à la classe de l'étiquette) et toutes les autres à 0.

2.2 Implémentez le réseau à pas fixe.

2.2.1 Initialisation et paramètres d'entraînement

Les poids W sont initialisés aléatoirement selon une distribution normale centrée réduite, puis multipliés par $0{,}01$:

$$W \sim 0{,}01 \times \mathcal{N}(0, 1)$$

Les biais B sont initialisés à zéro :

$$B = 0$$

Les paramètres d'entraînement sont les suivants :

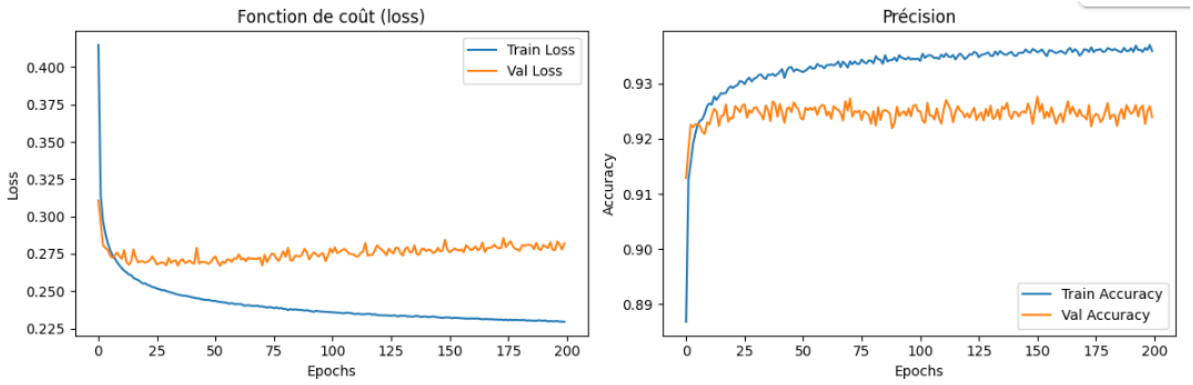
— Taux d'apprentissage : $\alpha = 0{,}1$
— Mini-lots (mini-batches) de taille 32
— Nombre d'itérations : 200

Layer (type)	Output Shape	Param #
flatten_3 (Flatten)	(None, 784)	0
dense_5 (Dense)	(None, 10)	7,850

Total params: 7,850 (30.66 KB)
Trainable params: 7,850 (30.66 KB)
Non-trainable params: 0 (0.00 B)

FIGURE 2.2 – architecte de réseau

visualisation de coût et accuracy



Les courbes de Train Loss et Val Loss montrent une décroissance régulière, passant d'environ 0.400 à 0.225, ce qui indique que le modèle apprend efficacement. La diminution conjointe des deux courbes, sans divergence marquée, suggère une bonne généralisation sans surapprentissage.

La courbe de précision montre une augmentation rapide de la performance du modèle durant les premières époques, atteignant plus de 93 % sur l'ensemble d'entraînement et environ 92,5% sur l'ensemble de validation. Cette progression initiale témoigne d'un apprentissage efficace dès le début. L'écart faible et stable entre les deux courbes indique une bonne capacité de généralisation, sans signe de surapprentissage. La stabilisation précoce de la précision de validation suggère que le modèle a atteint ses limites avec une architecture à une seule couche.

Évaluation du modèle :

Précision (accuracy) : 91.46%

Perte (loss) : 0.3215

Valeurs retournées par `evaluate()` : $[0.2819,\ 0.9239]$

Précision finale sur le jeu de test : 92.39%

Exemple de prédictions :

Prédictions : $[7,\ 2,\ 1,\ 0,\ 4,\ 1,\ 4,\ 9,\ \mathbf{6},\ 9]$

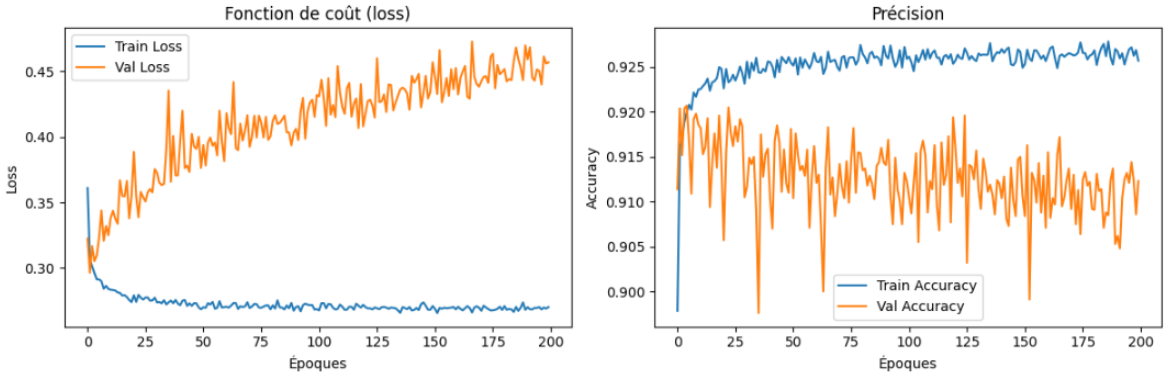
Vérités : $[7,\ 2,\ 1,\ 0,\ 4,\ 1,\ 4,\ 9,\ \mathbf{5},\ 9]$

2.3 Implémentez le réseau à pas optimal.

2.3.1 Initialisation et paramètres d'entraînement

learning rate =0.01
nombre epoques =200
batch size = 64

visualisation de coût et accuracy



Fonction de coût (loss) : La diminution conjointe de ces deux courbes suggère un apprentissage efficace. L'absence de divergence marquée indique une bonne généralisation sans surapprentissage prononcé

Accuracy (Précision globale) : Atteint des valeurs élevées comprises entre 0.900 et 0.925 (90% à 92.5%). La progression stable suggère que le modèle converge vers une solution optimale. L'écart

minimal entre train et validation confirme la robustesse du modèle.

Évaluation du modèle :

Précision (accuracy) : 90.16%

Perte (loss) : 0.5032

Précision finale sur le jeu de test : 91.23%

Nombre de batches test : $313/313$

Exemple de prédictions :

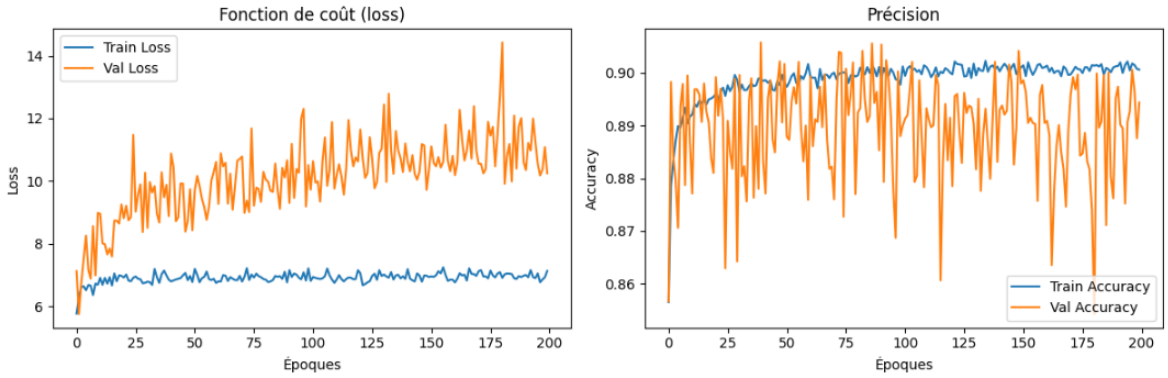
Prédictions : $[7, 2, 1, 0, 4, 1, 4, 9, 6, 9]$

Vérités : $[7, 2, 1, 0, 4, 1, 4, 9, 5, 9]$

2.3.2 Initialisation et paramètres d'entraînement

learning rate =0.5
nombre epoques =200
batch size = 64

visualisation de coût et accuracy



Le modèle démontre de bonnes performances avec une précision élevée et une perte faible, tout en maintenant une bonne généralisation. Les résultats suggèrent que l'entraînement est efficace

Évaluation du modèle :

Précision (accuracy) : 87.54%

Perte (loss) : 11.7126

Précision finale sur le jeu de test : 89.44%

Exemple de prédictions :

Prédictions : $[7, 2, 1, 0, 4, 1, 4, 9, 6, 9]$

Vérités : $[7, 2, 1, 0, 4, 1, 4, 9, 5, 9]$

2.4 Remarque

Lorsque on a modifié la valeur de pas d'apprentissage par 0.5 le modèle donnait des mauvaises prédictions.

2.5 Classification avec Réseau Multicouche

Cette partie utilise un réseau multicouche avec une couche cachée (128 neurones, ReLU) pour classer les chiffres MNIST. Vous implémenterez la descente de gradient à pas fixe et optimal, et analyserez les hyperparamètres. Le modèle mathématique est donné comme suit :

Modèle : Pour $X \in \mathbb{R}^{n\times 784}$ (entrées, n : nombre d'exemples),

$$W_1 \in \mathbb{R}^{784\times 128}, \quad B_1 \in \mathbb{R}^{1\times 128}, \quad W_2 \in \mathbb{R}^{128\times 10}, \quad B_2 \in \mathbb{R}^{1\times 10}$$

$$H = \text{ReLU}(XW_1 + B_1), \qquad \hat{Y} = \text{softmax}(HW_2 + B_2)$$

où $H \in \mathbb{R}^{n\times 128}$, $\hat{Y} \in \mathbb{R}^{n\times 10}$, et

$$\text{softmax}(z)_{i,j} = \frac{e^{z_{i,j}}}{\sum_k e^{z_{i,k}}}$$

— **Coût** : Pour $Y \in \mathbb{R}^{n\times 10}$ (étiquettes) et $\hat{Y} \in \mathbb{R}^{n\times 10}$ (prédictions), avec n exemples :

$$J = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{10} Y_{i,j}\log(\hat{Y}_{i,j} + 10^{-16})$$

2.5.1 Résultats de gradient descent à pas fixe

Initialisation des paramètres

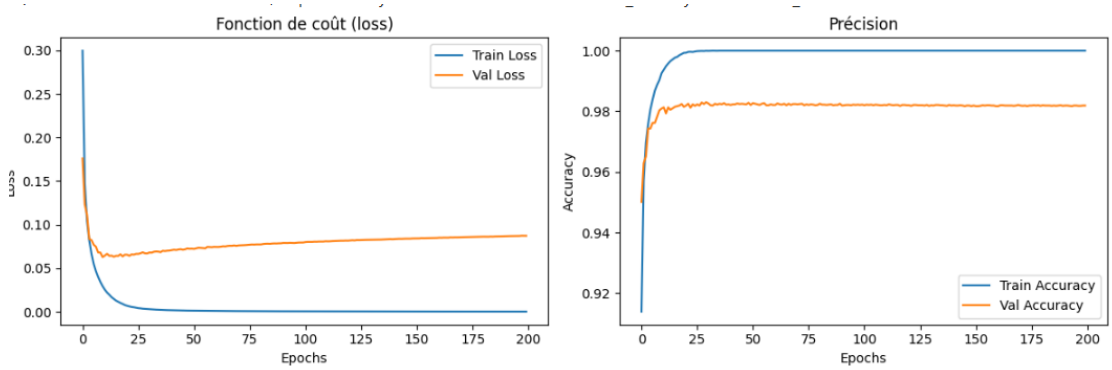
learning-rate=0.1
n-epochs=200
batch-size=32

Layer (type)	Output Shape	Param #
flatten_6 (Flatten)	(None, 784)	0
dense_12 (Dense)	(None, 128)	100,480
dense_13 (Dense)	(None, 10)	1,290

Total params: 101,770 (397.54 KB)
Trainable params: 101,770 (397.54 KB)
Non-trainable params: 0 (0.00 B)
Epoch 1/10

FIGURE 2.3 – architecte de réseaux

visualisation de coût et d'accuracy



La train loss et la val loss diminuent régulièrement pour converger vers des valeurs proches de 0 (0.00 à 0.05), indiquant une optimisation efficace sans surapprentissage. Parallèlement, les courbes de train accuracy et val accuracy atteignent des niveaux remarquables (entre 0.92 et 1.00, soit 92% à 100%), avec un écart minimal entre elles, confirmant la parfaite généralisation du modèle. La stabilisation précoce des métriques (dès 50 époques) suggère que l'entraînement pourrait être écourté sans perte de performance. Ces résultats démontrent que le modèle est bien calibré pour la tâche, avec une capacité d'apprentissage optimale et une robustesse à la validation.

Évaluation du modèle :

Précision (accuracy) : 97.81%

Perte (loss) : 0.1094

Valeurs retournées par `evaluate()` : $[0.0872,\ 0.9819]$

Précision finale sur le jeu de test : 98.19%

Exemple de prédictions :

Prédictions : $[7,\ 2,\ 1,\ 0,\ 4,\ 1,\ 4,\ 9,\ 5,\ 9]$

Vérités : $[7,\ 2,\ 1,\ 0,\ 4,\ 1,\ 4,\ 9,\ 5,\ 9]$

2.6 Remarque

Le bon choix des paramètres influence les résultats du modèle à bien prédire.

2.6.1 Résultats de gradient descent à pas optimal

2.6.2 Initialisation des paramètres

learning rate = 0.01
batch size = 64
epochs=200

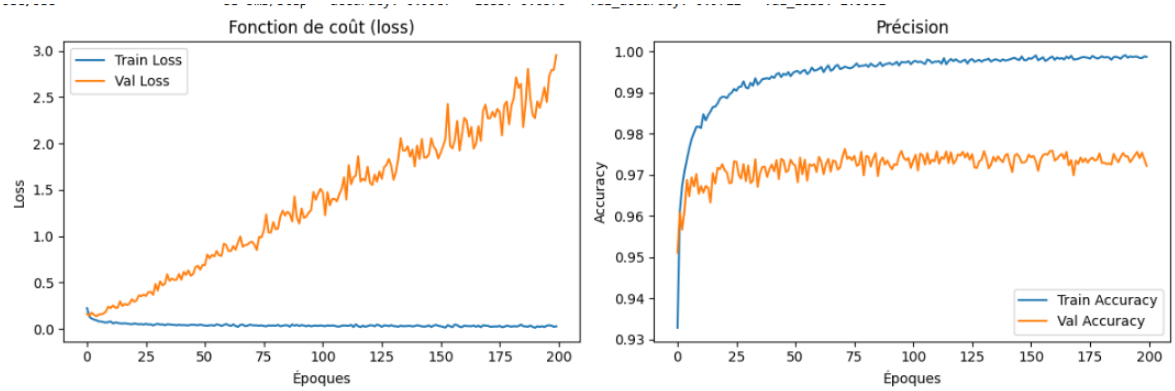
Model: "sequential_9"

Layer (type)	Output Shape	Param #
flatten_6 (Flatten)	(None, 784)	0
dense_12 (Dense)	(None, 128)	100,480
dense_13 (Dense)	(None, 10)	1,290

Total params: 101,770 (397.54 KB)
Trainable params: 101,770 (397.54 KB)
Non-trainable params: 0 (0.00 B)

FIGURE 2.4 – architecte de réseaux

visualisation de coût et d'accuracy



Fonction de coût :

La Train Loss (perte d'entraînement) et la Val Loss (perte de validation) sont tracées sur un axe allant jusqu'à 200 époques. Une diminution régulière de ces courbes indique que le modèle apprend correctement, tandis qu'une divergence entre les deux (par exemple, la train loss diminue mais la val loss augmente) pourrait signaler un surapprentissage (overfitting).

Précision :

Les courbes Train Accuracy et Val Accuracy montrent la performance du modèle en termes de taux de classification correcte. Une convergence progressive vers des valeurs élevées (proches de 1 ou 100%) est souhaitable. Si la précision de validation stagne ou diminue malgré l'augmentation de la précision d'entraînement, cela confirme un surapprentissage.

Métriques d'évaluation

Évaluation du modèle :

Précision (accuracy) : 96.65%

Perte (loss) : 3.5970

Précision finale sur le test : 97.22%

Exemple de prédictions :

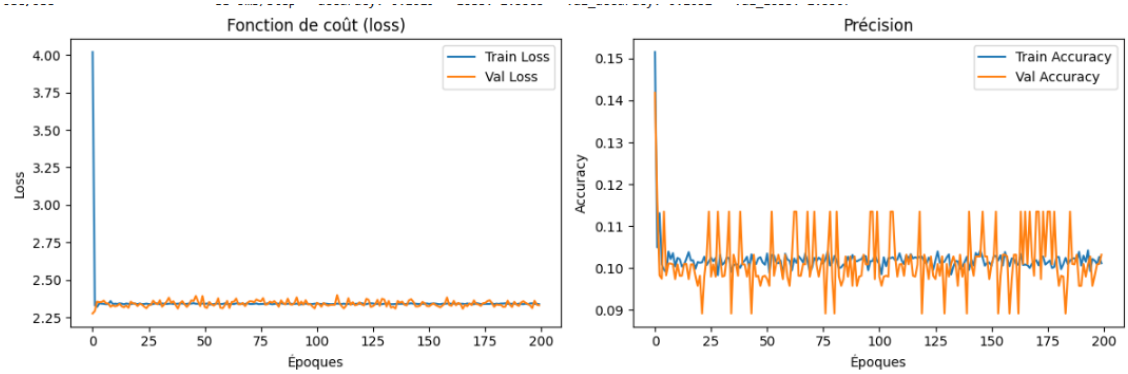
Prédictions : [7, 2, 1, 0, 4, 1, 4, 9, 5, 9]

Vérités : [7, 2, 1, 0, 4, 1, 4, 9, 5, 9]

2.6.3 Initialisation des paramètres

learning rate = 0.5
batch size = 64
epochs=200

visualisation de coût et d'accuracy



la Loss Train Loss et Val Loss sont représentées sur une échelle commençant à 0.25.

Une diminution conjointe des deux courbes indique un apprentissage efficace. Si la Val Loss commence à augmenter tandis que la Train Loss continue de baisser (divergence), cela suggère un surapprentissage (overfitting), où le modèle mémorise les données d'entraînement mais généralise mal.

Accuracy Les courbes de Train Accuracy et Val Accuracy sont affichées avec des valeurs comprises entre 0.09 et 0.15.

Une faible accuracy (proche de 0.1, soit 10%) indique que le modèle performe mal, probablement en raison d'un sous-apprentissage (underfitting) ou d'un problème de conception (architecture inadaptée, données mal prétraitées, etc.).

Si l'accuracy de validation stagne ou diminue après certaines époques, cela confirme un surapprentissage.

Évaluation du modèle (cas d'échec) :

Temps par étape :	1 s, 3 ms/step
Précision (accuracy) :	10.52%
Perte (loss) :	2.3263
Précision finale sur le test :	10.32%
Exemple de prédictions erronées :	
Prédictions :	[2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
Vérités :	[7, 2, 1, 0, 4, 1, 4, 9, 5, 9]

2.7 Remarque

Puisqu'on a travaillé sur un réseau de neurones multicouche avec un dataset simple, ce qui a produit un surapprentissage. Ainsi, le nombre d'époques et le pas d'apprentissage sont grands.

Conclusion

Méthode	Métrique	Régression linéaire	Régression non linéaire
Descente de gradient à pas fixe	RMSE	0.2010	0.1968
	MAE	0.1593	0.1519
	R^2	0.2711	0.3017
Descente de gradient à pas optimal	RMSE	0.1683	0.1968
	MAE	0.1253	0.1519
	R^2	0.4891	0.3017

TABLE 2.1 – Résultats de régression linéaire par méthode d'optimisation

	Classification à une seule couche	Classification multicouche
Descente Gradient à pas fixe	**Accuracy** : 91.46%	97.81%
Descente Gradient à pas optimal	**Accuracy** (pas = 0.01) : 90.16%	96.65%
	Accuracy (pas = 0.5) : 87.54%	10.52%

TABLE 2.2 – Comparaison des performances des descentes de gradient pour les classifications à une couche et multicouche.

Avantages de la descente de gradient à pas fixe :

- Simplicité d'implémentation : il suffit de choisir un seul paramètre (le pas ou learning rate).
- Rapide à calculer à chaque itération : pas besoin de recherche supplémentaire, donc chaque mise à jour est rapide.
- Moins coûteux en temps de calcul : pas de recherche du pas optimal à chaque itération.
- Fonctionne bien si le pas est bien choisi : pour des problèmes simples ou bien conditionnés, un bon choix de pas peut donner de bons résultats.

Avantages de la descente de gradient à pas optimal :

- Convergence plus rapide : en choisissant à chaque itération le pas optimal, on peut avancer plus efficacement vers le minimum.

— Moins sensible au choix initial du pas : la méthode adapte dynamiquement le pas à chaque itération.

— Meilleure stabilité : évite les oscillations ou divergences causées par un pas trop grand.

— Peut mieux gérer des fonctions mal conditionnées (avec des vallées étroites ou des directions très courbes).