

Cours Machine Learning

Chapitre 03 Régression

Objectifs d'apprentissage :

- Maîtriser la régression linéaire (simple, multiple, polynomiale)
- Comprendre les techniques de régularisation (Ridge, Lasso, Elastic Net)
- Diagnostiquer et valider un modèle de régression
- Implémenter et optimiser des modèles de régression

Prérequis : Chapitres 00, 01, 02

Durée estimée : 5-6 heures

Notebooks : 03_demo_*.ipynb, 03_exercices.ipynb

Table des matières

1	Introduction à la Régression	2
1.1	Problématique	2
1.2	Fonction de Coût	2
2	Régression Linéaire Simple	3
2.1	Modèle	3
2.2	Solution Analytique	3
2.3	Coefficient de Détermination R^2	3
3	Régression Linéaire Multiple	4
3.1	Modèle Vectoriel	4
3.2	Solution par Moindres Carrés	4
3.3	Implémentation	5
4	Régression Polynomiale	5
4.1	Motivation	5
4.2	Exemple	6
5	Régularisation	6
5.1	Ridge Regression (L^2)	7
5.2	Lasso Regression (L^1)	7
5.3	Elastic Net	7
5.4	Comparaison Ridge vs Lasso	8
5.5	Choix de λ (Hyperparamètre)	8
6	Diagnostic et Analyse des Résidus	9
6.1	Résidus	9
6.2	Graphiques de Diagnostic	9
6.3	Multicollinéarité	9
7	Validation et Sélection de Modèle	9
7.1	Train/Validation/Test Split	9
7.2	Validation Croisée (Cross-Validation)	10
7.3	Courbe d'Apprentissage	10
8	Extensions et Variantes	10
8.1	Régression Robuste	10
8.2	Régression Non-Linéaire	10
8.3	Régression Généralisée (GLM)	11
9	Résumé du Chapitre	11
9.1	Points Clés	11
9.2	Formules Essentielles	11
10	Exercices	12
10.1	Questions de Compréhension	12

10.2 Exercices Pratiques	12
11 Pour Aller Plus Loin	12
11.1 Lectures Recommandées	12
11.2 Prochaines Étapes	12

1 Introduction à la Régression

1.1 Problématique

Définition : Régression

La régression est une tâche d'apprentissage supervisé où l'objectif est de prédire une variable cible **continue** $y \in \mathbb{R}$ à partir de features $\mathbf{x} \in \mathbb{R}^d$.

Formulation mathématique :

Étant donné un dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, on cherche une fonction $f : \mathbb{R}^d \rightarrow \mathbb{R}$ telle que :

$$\hat{y}_i = f(\mathbf{x}_i) \approx y_i \quad (1)$$

Différence avec la classification :

- **Régression** : y est continu (ex : prix, température, âge)
- **Classification** : y est discret (ex : spam/non-spam, classe 0/1/2)

Exemple : Applications de la régression

- **Immobilier** : Prédire le prix d'une maison (surface, nb pièces, localisation)
- **Finance** : Prédire le cours d'une action, le risque de défaut
- **Santé** : Prédire la durée d'hospitalisation, la progression d'une maladie
- **Marketing** : Prédire les ventes futures, le chiffre d'affaires
- **Météo** : Prédire la température, les précipitations

1.2 Fonction de Coût

La fonction de coût (loss) mesure l'erreur de prédiction. Pour la régression, les plus courantes sont :

1. Mean Squared Error (MSE) :

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (2)$$

2. Mean Absolute Error (MAE) :

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (3)$$

3. Root Mean Squared Error (RMSE) :

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (4)$$

Astuce**Choix de la métrique :**

- **MSE/RMSE** : Plus sensibles aux outliers (erreurs au carré), différentiables
- **MAE** : Plus robuste aux outliers, moins sensible aux valeurs extrêmes
- En pratique, MSE est la plus utilisée car différentiable et optimisable analytiquement

2 Régression Linéaire Simple

2.1 Modèle

Définition : Régression linéaire simple

Modèle avec une seule feature x :

$$\hat{y} = w_1x + w_0 \quad (5)$$

où w_1 est la pente et w_0 l'intercept (ordonnée à l'origine).

Notation alternative : $\hat{y} = \beta_1x + \beta_0$ ou $\hat{y} = ax + b$

2.2 Solution Analytique

Pour minimiser le MSE, on résout :

$$\min_{w_0, w_1} \sum_{i=1}^n (y_i - w_1x_i - w_0)^2 \quad (6)$$

En dérivant par rapport à w_0 et w_1 et en annulant, on obtient les **équations normales** :

$$w_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} = \frac{\text{Cov}(x, y)}{\text{Var}(x)} \quad (7)$$

$$w_0 = \bar{y} - w_1\bar{x} \quad (8)$$

où \bar{x} et \bar{y} sont les moyennes de x et y .

2.3 Coefficient de Détermination R^2

Définition : Coefficient R^2

Le coefficient de détermination mesure la proportion de variance expliquée par le modèle :

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}} \quad (9)$$

Interprétation :

- $R^2 = 1$: Modèle parfait (toutes les prédictions exactes)

- $R^2 = 0$: Modèle aussi bon qu'une prédiction constante ($\hat{y} = \bar{y}$)
- $R^2 < 0$: Modèle pire que la moyenne (rare, indique un problème)

Relation avec la corrélation : Pour la régression simple, $R^2 = \rho^2$ où ρ est le coefficient de corrélation de Pearson.

3 Régression Linéaire Multiple

3.1 Modèle Vectoriel

Définition : Régression linéaire multiple

Modèle avec d features :

$$\hat{y} = w_1x_1 + w_2x_2 + \cdots + w_dx_d + w_0 = \mathbf{w}^T \mathbf{x} + w_0 \quad (10)$$

En notation matricielle, pour n instances :

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} \quad (11)$$

où $\mathbf{X} \in \mathbb{R}^{n \times (d+1)}$ inclut une colonne de 1 pour l'intercept, et $\mathbf{w} \in \mathbb{R}^{d+1}$ contient tous les poids.

3.2 Solution par Moindres Carrés

Problème d'optimisation :

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 \quad (12)$$

Théorème : Équations normales

La solution optimale des moindres carrés est :

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (13)$$

sous réserve que $\mathbf{X}^T \mathbf{X}$ soit inversible (rang plein).

Démonstration (esquisse) :

$$\text{Minimiser } f(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 \quad (14)$$

$$\nabla_{\mathbf{w}} f = 2\mathbf{X}^T(\mathbf{X}\mathbf{w} - \mathbf{y}) = \mathbf{0} \quad (15)$$

$$\mathbf{X}^T \mathbf{X}\mathbf{w} = \mathbf{X}^T \mathbf{y} \quad (16)$$

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (17)$$

Attention

Problèmes numériques :

- Si $\mathbf{X}^T \mathbf{X}$ est singulière ou mal conditionnée, l'inversion est instable
- Causes : multicollinearité (features corrélées), $n < d$ (plus de features que d'ins-

tances)
 — Solutions : régularisation (Ridge, Lasso), sélection de features, PCA

3.3 Implémentation

```

1 from sklearn.linear_model import LinearRegression
2 from sklearn.model_selection import train_test_split
3 from sklearn.metrics import mean_squared_error, r2_score
4 import numpy as np
5
6 # Données
7 X = np.random.randn(100, 5) # 100 instances, 5 features
8 y = 3*X[:, 0] + 2*X[:, 1] - X[:, 2] + np.random.randn(100)*0.5
9
10 # Split train/test
11 X_train, X_test, y_train, y_test = train_test_split(
12     X, y, test_size=0.2, random_state=42
13 )
14
15 # Entraînement
16 model = LinearRegression()
17 model.fit(X_train, y_train)
18
19 # Prédition
20 y_pred = model.predict(X_test)
21
22 # Évaluation
23 mse = mean_squared_error(y_test, y_pred)
24 r2 = r2_score(y_test, y_pred)
25
26 print(f"MSE: {mse:.4f}")
27 print(f"R²: {r2:.4f}")
28 print(f"Coefficients: {model.coef_}")
29 print(f"Intercept: {model.intercept_.4f}")

```

Listing 1 – Régression linéaire avec scikit-learn

4 Régression Polynomiale

4.1 Motivation

La régression linéaire suppose une relation linéaire entre features et cible. Si la relation est non-linéaire, on peut utiliser des features polynomiales.

Définition : Régression polynomiale

Pour une feature x , on crée des features polynomiales :

$$\hat{y} = w_0 + w_1x + w_2x^2 + w_3x^3 + \cdots + w_px^p \quad (18)$$

Le modèle reste linéaire en les poids \mathbf{w} , mais non-linéaire en x .

Généralisation à plusieurs features : Pour d features, on peut créer tous les termes de degré $\leq p$:

$$\phi(\mathbf{x}) = [1, x_1, x_2, x_1^2, x_1x_2, x_2^2, \dots] \quad (19)$$

Le nombre de features polynomiales croît combinatoirement : $\binom{d+p}{p}$.

4.2 Exemple

Données 1D non-linéaires : $y = \sin(2\pi x) + \epsilon$

- **Degré 1 (linéaire)** : sous-ajustement (underfitting)
- **Degré 3-5** : bon ajustement
- **Degré 15** : sur-ajustement (overfitting)

```

1 from sklearn.preprocessing import PolynomialFeatures
2 from sklearn.pipeline import make_pipeline
3
4 # Créer un pipeline : features polynomiales + régression
5 degree = 3
6 model = make_pipeline(
7     PolynomialFeatures(degree=degree),
8     LinearRegression()
9 )
10
11 model.fit(X_train, y_train)
12 y_pred = model.predict(X_test)

```

Listing 2 – Régression polynomiale

Attention

Risque d'overfitting :

- Augmenter le degré p améliore le fit sur le train set
- Mais peut dégrader la généralisation (test set)
- Solution : validation croisée pour choisir p , régularisation

5 Régularisation

La régularisation pénalise la complexité du modèle pour éviter l'overfitting.

5.1 Ridge Regression (L^2)

Définition : Ridge (Régression L^2)

Ajoute une pénalité sur la norme L^2 des poids :

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} \left\{ \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 \right\} \quad (20)$$

où $\lambda \geq 0$ est le coefficient de régularisation.

Solution analytique :

$$\mathbf{w}_{\text{Ridge}}^* = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y} \quad (21)$$

Propriétés :

- Pénalise les poids de grande magnitude
- Réduit l'overfitting
- Solution toujours unique (même si $\mathbf{X}^T \mathbf{X}$ singulière)
- Les poids sont **rétrécis** (shrinkage) mais rarement nuls

5.2 Lasso Regression (L^1)

Définition : Lasso (Régression L^1)

Ajoute une pénalité sur la norme L^1 des poids :

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} \left\{ \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_1 \right\} \quad (22)$$

Propriétés :

- Pénalise les poids en valeur absolue
- Effectue une **sélection de features automatique** : certains poids deviennent exactement 0
- Pas de solution analytique simple, résolu par optimisation (coordinate descent, proximal gradient)
- Utile quand on suspecte que peu de features sont pertinentes

5.3 Elastic Net

Définition : Elastic Net

Combinaison de Ridge et Lasso :

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} \left\{ \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda_1 \|\mathbf{w}\|_1 + \lambda_2 \|\mathbf{w}\|_2^2 \right\} \quad (23)$$

ou avec un ratio $\rho \in [0, 1]$:

$$\text{Pénalité} = \lambda \left[\rho \|\mathbf{w}\|_1 + \frac{1 - \rho}{2} \|\mathbf{w}\|_2^2 \right] \quad (24)$$

Avantages :

- Combine les avantages de Ridge et Lasso
- Sélection de features (comme Lasso)
- Stabilité quand features corrélées (comme Ridge)

5.4 Comparaison Ridge vs Lasso

TABLE 1 – Ridge vs Lasso

Aspect	Ridge (L^2)	Lasso (L^1)
Pénalité	$\lambda \sum w_i^2$	$\lambda \sum w_i $
Solution analytique	Oui	Non
Sélection de features	Non	Oui
Poids à zéro	Jamais exactement	Oui
Features corrélées	Stable	Instable (choix arbitraire)
Interprétabilité	Moyenne	Élevée (sparse)

Astuce

Quand utiliser quoi ?

- **Ridge** : Toutes les features potentiellement utiles, features corrélées
- **Lasso** : Peu de features importantes, besoin d'interprétabilité
- **Elastic Net** : Compromis, features corrélées + sélection

5.5 Choix de λ (Hyperparamètre)

Le coefficient de régularisation λ contrôle le trade-off biais-variance :

- $\lambda = 0$: Pas de régularisation (risque d'overfitting)
- λ petit : Régularisation faible
- λ grand : Régularisation forte (risque d'underfitting)

Méthode de sélection : Validation croisée

```

1 from sklearn.linear_model import RidgeCV
2
3 # Tester plusieurs valeurs de lambda (alpha en scikit-learn)
4 alphas = np.logspace(-3, 3, 50)
5 model = RidgeCV(alphas=alphas, cv=5)
6 model.fit(X_train, y_train)
7
8 print(f"Meilleur alpha: {model.alpha_:.4f}")

```

Listing 3 – Ridge avec validation croisée

6 Diagnostic et Analyse des Résidus

6.1 Résidus

Définition : Résidus

Les résidus sont les erreurs de prédiction :

$$e_i = y_i - \hat{y}_i \quad (25)$$

Hypothèses de la régression linéaire :

1. **Linéarité** : Relation linéaire entre X et y
2. **Indépendance** : Les résidus sont indépendants
3. **Homoscédasticité** : Variance constante des résidus
4. **Normalité** : Les résidus suivent une loi normale
5. **Absence de multicollinéarité** : Features non (trop) corrélées

6.2 Graphiques de Diagnostic

1. Résidus vs Prédictions :

- Vérifier l'homoscédasticité
- Pattern : résidus dispersés aléatoirement autour de 0
- Problème : tendance, forme en entonnoir (hétéroscléasticité)

2. Q-Q Plot :

- Vérifier la normalité des résidus
- Points alignés sur la diagonale = normalité

3. Résidus vs Features :

- Déetecter des relations non-linéaires manquées

6.3 Multicollinéarité

Définition : Multicollinéarité

Corrélation forte entre plusieurs features. Rend l'estimation des coefficients instable.

Détection :

- Matrice de corrélation : $|\rho_{ij}| > 0.8$ problématique
- VIF (Variance Inflation Factor) : $VIF > 10$ indique multicollinéarité

Solutions :

- Supprimer une des features corrélées
- PCA (réduction de dimensionnalité)
- Régularisation (Ridge)

7 Validation et Sélection de Modèle

7.1 Train/Validation/Test Split

Procédure standard :

1. **Train set (70%)** : Entraîner le modèle
2. **Validation set (15%)** : Tuner les hyperparamètres
3. **Test set (15%)** : Évaluation finale (une seule fois)

7.2 Validation Croisée (Cross-Validation)

Définition : K-Fold Cross-Validation

Diviser les données en K folds. Pour chaque fold :

- Entraîner sur $K - 1$ folds
- Valider sur le fold restant
- Répéter K fois, moyenner les scores

```

1 from sklearn.model_selection import cross_val_score
2
3 model = Ridge(alpha=1.0)
4 scores = cross_val_score(model, X, y, cv=5,
5                           scoring='neg_mean_squared_error')
6 rmse_scores = np.sqrt(-scores)
7
8 print(f"RMSE: {rmse_scores.mean():.4f} (+/- {rmse_scores.std():.4f})")

```

Listing 4 – Validation croisée

Avantages :

- Utilise toutes les données pour train et validation
- Estimation plus robuste de la performance
- Réduit le risque de chance (lucky/unlucky split)

7.3 Courbe d'Apprentissage

Tracer la performance (MSE, R^2) en fonction de la taille du train set.

Diagnostic :

- **Underfitting** : Train et validation scores bas et proches
- **Overfitting** : Grand écart entre train et validation scores
- **Bon fit** : Scores convergent vers une valeur élevée

8 Extensions et Variantes

8.1 Régression Robuste

Pour gérer les outliers :

- **Huber Regression** : Loss hybride (MSE + MAE)
- **RANSAC** : Fit sur inliers, ignore outliers

8.2 Régression Non-Linéaire

Au-delà des polynômes :

- **Kernel Ridge Regression** : Kernels (RBF, polynomial)
- **Support Vector Regression (SVR)** : SVM pour la régression
- **Decision Trees, Random Forests** : Modèles non-linéaires
- **Gradient Boosting (XGBoost, LightGBM)** : État de l'art pour données tabulaires
- **Réseaux de neurones** : Deep Learning pour régression

8.3 Régression Généralisée (GLM)

Extension de la régression linéaire pour distributions non-gaussiennes :

- Régression logistique (Bernoulli)
- Régression de Poisson
- Régression Gamma

9 Résumé du Chapitre

9.1 Points Clés

- **Régression linéaire** : Modèle simple, interprétable, solution analytique
- **Équations normales** : $\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$
- R^2 : Mesure de qualité du fit (0 à 1)
- **Régression polynomiale** : Capture des relations non-linéaires
- **Ridge (L^2)** : Régularisation, stabilité
- **Lasso (L^1)** : Sélection de features, sparsité
- **Diagnostic** : Analyse des résidus, détection de problèmes
- **Validation croisée** : Sélection d'hyperparamètres robuste

9.2 Formules Essentielles

Formules à retenir

Régression linéaire simple :

$$w_1 = \frac{\text{Cov}(x, y)}{\text{Var}(x)}, \quad w_0 = \bar{y} - w_1 \bar{x} \quad (26)$$

Régression multiple (équations normales) :

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (27)$$

Ridge :

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y} \quad (28)$$

Coefficient R^2 :

$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2} \quad (29)$$

10 Exercices

10.1 Questions de Compréhension

1. Quelle est la différence entre régression et classification ?
2. Pourquoi utilise-t-on le MSE plutôt que le MAE en régression linéaire ?
3. Expliquez pourquoi la régression polynomiale de degré élevé peut causer de l'overfitting.
4. Quelle est la différence principale entre Ridge et Lasso ?
5. Comment détecte-t-on la multicollinéarité ? Quelles sont les solutions ?
6. Pourquoi la validation croisée est-elle préférable à un simple train/test split ?

10.2 Exercices Pratiques

Voir le notebook `03_exercices.ipynb` (*solutions intégrées*)

11 Pour Aller Plus Loin

11.1 Lectures Recommandées

- *An Introduction to Statistical Learning* (2e éd., 2021) - James et al. (Ch. 3)
- *The Elements of Statistical Learning* (2009) - Hastie et al. (Ch. 3)
- Scikit-learn documentation : Linear Models

11.2 Prochaines Étapes

Chapitre suivant : **Chapitre 04 - Classification Supervisée**

Références

1. Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer.
2. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning* (2e éd.). Springer.
3. Géron, A. (2022). *Hands-On Machine Learning* (3e éd.). O'Reilly.