Chapitre 2 : Modèles Linéaires

Plongée au cœur de la modélisation discriminante

Plan

1. **La Fondation :** Régression Linéaire
2. **La Solution :** Équation Normale vs Méthodes Itératives
3. **Le Moteur :** Descente de Gradient & SGD
4. **Le Liant :** Polynômes & Ingénierie des Caractéristiques
5. **Le Contrôle :** Régularisation (Ridge, Lasso, Elastic Net)
6. **La Classification :** Régression Logistique & Softmax

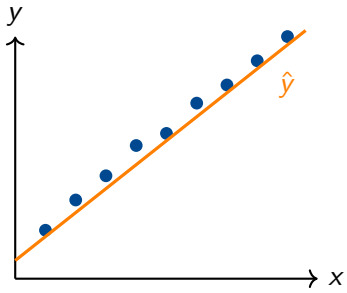
La Fondation

Configuration de la Régression Linéaire

L'Hypothèse la Plus Simple

En sciences et en ingénierie, la façon la plus simple de modéliser une relation est une ligne droite.

$$y = mx + b$$



Le Problème de Prédiction

- **Entrée :** Un vecteur de caractéristiques (features) x.
  - Exemple : [Superficie, Nombre de chambres, Âge de la maison]
- **Cible :** Un nombre continu y.
  - Exemple : Prix de la maison.
- **But :** Trouver une fonction $h(x)$ qui produit $\hat{y}$ proche de y.

L'Hypothèse $h_\theta(x)$

Nous définissons notre "hypothèse" (notre modèle) comme une somme pondérée des entrées.

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

- θ_0 : Le **Biais** (ou Intercepte). La valeur de $\hat{y}$ quand tous les x sont à 0.
- θ_i : Le **Poids** (ou Coefficient) pour la caractéristique i.
- x_i : La valeur de la caractéristique d'entrée.

Vectorisation (1/2)

Écrire de longues sommes est fastidieux et lent informatiquement. Nous utilisons l'Algèbre Linéaire.

L'Astuce du Biais : Nous ajoutons une caractéristique "fictive" $x_0 = 1$.

- $\theta = [\theta_0, \theta_1, \ldots, \theta_n]^T$
- $x = [1, x_1, \ldots, x_n]^T$

Maintenant, les deux vecteurs ont une taille $n + 1$.

Vectorisation (2/2) : Le Produit Scalaire

Maintenant, nous pouvons écrire l'hypothèse comme un simple Produit Scalaire :

$$\boxed{\hat{y} = h_\theta(x) = \theta^T x}$$

C'est la notation standard que vous verrez dans les articles de recherche.

Quelle est la Qualité du Modèle ?

Supposons que nous choisissons des poids θ aléatoires. La ligne sera terrible.

Nous avons besoin d'un moyen de mesurer à quel point le modèle est mauvais.

Fonction de Coût (Loss Function)

Notée $J(\theta)$, elle quantifie l'erreur du modèle.

Pour un seul exemple, l'erreur est :

$$\text{Erreur} = \hat{y}^{(i)} - y^{(i)}$$

La Fonction de Coût : MSE

Nous utilisons l'Erreur Quadratique Moyenne (MSE) :

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(\theta^T x^{(i)} - y^{(i)} \right)^2$$

Pourquoi au Carré ?

- Cela transforme les erreurs négatives en positives.
- Cela pénalise les grandes erreurs beaucoup plus que les petites.
  - Erreur de $2 \rightarrow 4$; Erreur de $10 \rightarrow 100$.

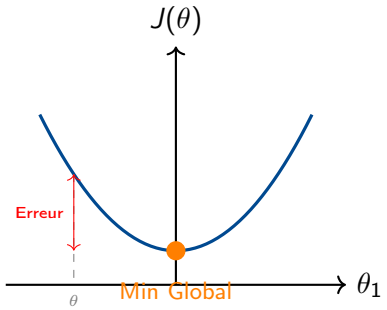
Géométrie de la MSE

La fonction MSE crée une surface dans l'espace des paramètres.

Pour la Régression Linéaire, cette surface est un **Bol Convexe**.

Propriété Clé

Un seul minimum global. Pas de minima locaux.



L'Approche Analytique

Résoudre la Régression Linéaire : L'Approche Analytique

Le But

Nous voulons trouver le vecteur θ qui minimise $J(\theta)$.

Puisque $J(\theta)$ est une fonction quadratique lisse, nous pouvons utiliser le calcul différentiel.

Principe

Au point minimum, la pente (gradient) est nulle : $\nabla_\theta J(\theta) = 0$

L'Équation Normale

En posant $\nabla_\theta J(\theta) = 0$ et en résolvant pour θ, nous obtenons :

$$\boxed{\hat{\theta} = (X^T X)^{-1} X^T y}$$

Cette formule nous donne les poids optimaux **immédiatement**.

La Matrice de Design X

Forme : $(m \times (n+1))$

- m = nombre d'exemples d'entraînement (lignes).
- $n+1$ = nombre de caractéristiques + biais (colonnes).

$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots \\ \vdots & \vdots & \vdots & \ddots \\ 1 & x_1^{(m)} & x_2^{(m)} & \dots \end{bmatrix}$$

La colonne 1 est remplie de 1 (pour le biais θ_0).

Le Goulot d'Étranglement Informatique

L'Équation Normale est mathématiquement belle, mais dangereuse informatiquement.

$$\hat{\theta} = \underbrace{(X^T X)^{-1}}_{\text{Inverse}} X^T y$$

- Le terme $(X^T X)$ est une matrice de taille $(n+1) \times (n+1)$.
- Calculer l'Inverse a une complexité de $O(n^3)$.

Problème

$n = 100,000$ caractéristiques $\rightarrow 10^{15}$ opérations. **Impossible !**

Le Moteur

Optimisation via Descente de Gradient

La Solution Itérative

Si nous ne pouvons pas le résoudre analytiquement, nous le résolvons **itérativement**.

La Descente de Gradient est le **pilier fondamental** du machine learning moderne et du deep learning.

L'Intuition : La Montagne dans le Brouillard

Imaginez que vous êtes sur une montagne, de nuit, dans le brouillard.

- Vous voulez atteindre la vallée la plus basse (Coût Minimum).
- Vous ne pouvez pas voir le fond.
- Vous ne pouvez sentir que la pente sous vos pieds.

La Stratégie :

1. Sentez la pente.
2. Si le sol descend vers la droite, faites un pas vers la droite.
3. Répétez jusqu'à ce que la pente soit plate (0).

Les Maths : Le Gradient

La “pente” en plusieurs dimensions s’appelle le **Gradient**, noté $\nabla_\theta J(\theta)$.

- C’est un vecteur qui pointe dans la direction de la montée la plus raide.
- Pour descendre, nous allons dans la direction opposée ($-\nabla$).

La Formule de Mise à Jour

Algorithme :

1. Initialiser θ aléatoirement.
2. Répéter jusqu'à convergence :

$$\boxed{\theta_j := \theta_j - \eta \frac{\partial}{\partial \theta_j} J(\theta)}$$

- $:=$ signifie "mettre à jour la valeur".
- η (Eta) est le **Taux d'Apprentissage** (Learning Rate).

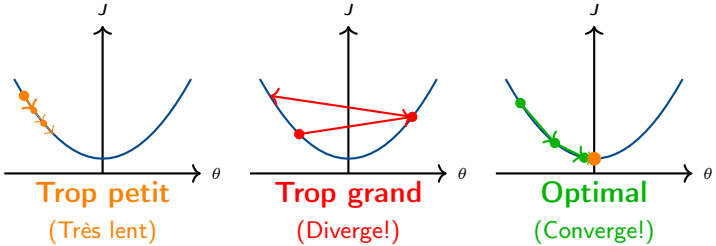
Calculer la Dérivée

Pour la MSE, la dérivée est :

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{2}{m} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)}) x_j^{(i)}$$

C'est l'erreur moyenne multipliée par la valeur d'entrée.

Le Taux d'Apprentissage (η) - Visualisation



Variations de la Descente de Gradient

Variations de la Descente de Gradient

Le Problème des Données

La formule du gradient nécessite de sommer sur m exemples.

$$\sum_{i=1}^{m} \dots$$

Si $m = 1,000,000,000$ (Big Data), calculer une étape prend une éternité.

Variante 1 : Batch Gradient Descent

Méthode : Utiliser TOUS les points de données pour calculer le gradient pour une étape.

Avantages :

- Stable, convergence fluide
- Solution optimale garantie (convexe)

Inconvénients :

- Extrêmement lent par étape
- Lourd en calcul

Variante 2 : Stochastic Gradient Descent (SGD)

Méthode : Choisir UN exemple aléatoire $(x^{(i)}, y^{(i)})$, calculer le gradient, mettre à jour.

$$\theta := \theta - \eta \nabla J(\theta; x^{(i)}, y^{(i)})$$

Avantages :

- Mises à jour ultra-rapides
- Peu de mémoire
- Apprentissage sur flux de données (temps réel)

Inconvénients :

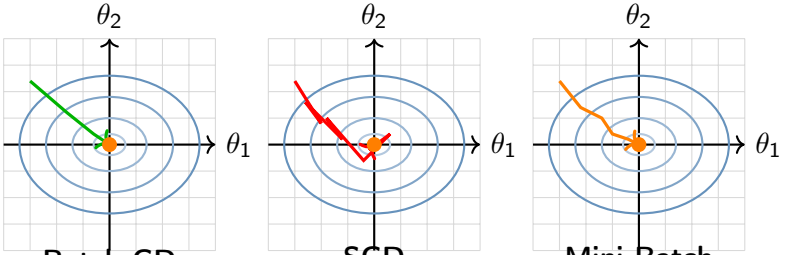
- Bruité ! Chemin "ivre"
- Oscille autour du minimum

Variante 3 : Mini-Batch (Le Standard)

Méthode : Choisir un petit Lot d'exemples (ex : 32, 64 ou 128).

- Plus rapide que Batch GD, plus stable que SGD.
- Permet les **Opérations Matricielles** (accélération GPU).

Comparaison des Trajectoires de Convergence



Le Liant

Ingénierie des Caractéristiques & Polynômes

La Limite de Linéarité

La Régression Linéaire suppose que les données suivent une ligne droite.

Question : Et si les données sont courbées ? (trajectoire d'une balle, croissance bactérienne)

Problème

Ajuster une ligne droite à une courbe résulte en un **Sous-apprentissage** (Underfitting).

L'Astuce Polynomiale

Nous n'avons pas besoin d'un nouvel algorithme. Nous avons simplement besoin de **nouvelles caractéristiques**.

Caractéristique Originale : x (Taille de la maison).

Nous créons :

- $x_1 = x$
- $x_2 = x^2$
- $x_3 = x^3$

$$\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

Linéaire par rapport à θ

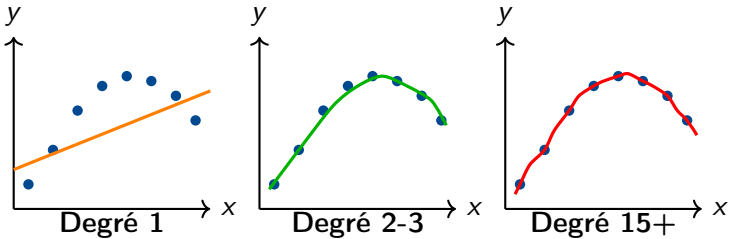
Attendez, n'est-ce pas non-linéaire ?

- C'est **non-linéaire** par rapport à x.
- Mais c'est **Linéaire** par rapport à θ.

Bonne Nouvelle

Nous pouvons toujours utiliser l'Équation Normale ou la Descente de Gradient !

Le Danger des Polynômes - Visualisation



Attention

Plus le modèle est flexible, plus il risque de mémoriser le bruit !

Le Contrôle

Biais, Variance et Régularisation

Généralisation du Modèle

Notre but n'est pas de mémoriser les données d'entraînement.

Objectif

Notre but est de **Généraliser** à de nouvelles données non vues.

Les Trois États de la Modélisation

Sous-apprentissage
(Biais Élevé)

- Modèle trop simple
- Rate la tendance
- Erreur Train/Test : Élevée

Surapprentissage
(Variance Élevée)

- Modèle trop complexe
- Mémorise le bruit
- Train : Faible, Test : Élevée

Juste Ce Qu'il Faut

- Capture le signal
- Ignore le bruit
- Erreur Train/Test : Faible

L'Intuition de la Régularisation

Le surapprentissage se produit quand les poids (θ) deviennent très grands.

Idée : Pénalisons le modèle s'il a de grands poids.

$$\boxed{J(\theta) = \text{MSE}(\theta) + \alpha \cdot \text{Pénalité}(\theta)}$$

α contrôle l'intensité de la régularisation.

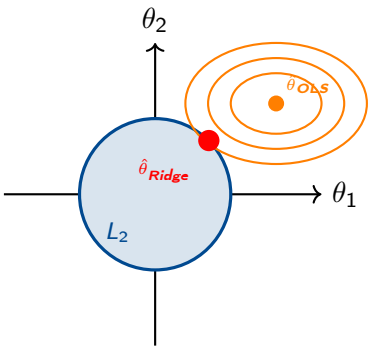
Type 1 : Régression Ridge (L_2)

Nous pénalisons la somme des **Carrés** des poids.

$$J(\theta) = \text{MSE} + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

Propriétés :

- Rétrécit les poids vers zéro uniformément.
- Poids petits, mais rarement exactement 0.



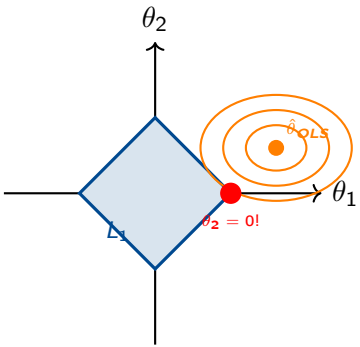
Type 2 : Régression Lasso (L_1)

Nous pénalisons les valeurs **Absolues**.

$$J(\theta) = \text{MSE} + \alpha \sum_{i=1}^{n} |\theta_i|$$

Propriétés :

- Force certains poids à **exactement 0**.
- Sélection de caractéristiques.



Type 3 : Elastic Net

Pourquoi choisir ? Utilisons les deux.

$$J(\theta) = \text{MSE} + r\alpha \sum |\theta_i| + \frac{1-r}{2}\alpha \sum \theta_i^2$$

- r : Ratio de mélange.
- Combine la **stabilité de Ridge** avec la **sélection de caractéristiques de Lasso**.

Classification

Régression Logistique

Changer le Problème

Jusqu'à présent, nous prédisions un nombre ($\hat{y} \in \mathbb{R}$).

Maintenant, nous voulons prédire une **Classe** ($y \in \{0, 1\}$).

- Spam vs Non Spam.
- Malin vs Bénin.
- Chat vs Chien.

Pourquoi la Régression Linéaire Échoue Ici

Rappel : Pour classifier, on utilise généralement un seuil (ex : 0.5) sur la prédiction.

Problèmes avec la régression linéaire :

- La prédiction $\hat{y}$ peut être > 1 ou < 0.
- Que signifie une prédiction de 2.5 ? Ou -0.3 ?
- Sensibilité aux valeurs aberrantes : une valeur extrême déplace le seuil de décision.

Solution

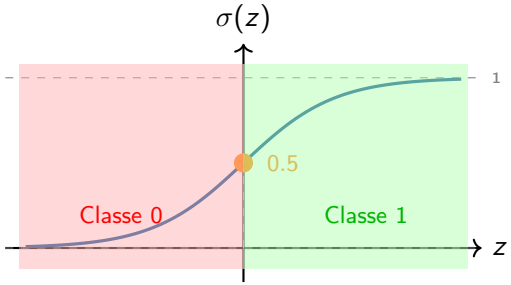
Nous voulons que notre sortie soit une **Probabilité** entre 0 et 1.

La Fonction Sigmoïde

Nous avons besoin d'une fonction qui mappe $(-\infty, \infty) \to (0, 1)$.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- $z = 0 \Rightarrow \sigma = 0.5$
- $z \to \infty \Rightarrow \sigma \to 1$
- $z \to -\infty \Rightarrow \sigma \to 0$



L'Hypothèse Logistique

Nous prenons notre fonction linéaire $\theta^T x$ et l'enveloppons dans la Sigmoïde.

$$\hat{p} = h_\theta(x) = \sigma(\theta^T x)$$

Sortie : La probabilité que $y = 1$.

Décision :

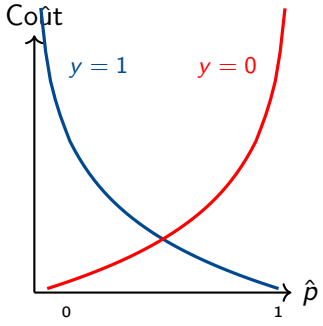
- Si $\hat{p} \geq 0.5$: Prédire Classe 1.
- Si $\hat{p} < 0.5$: Prédire Classe 0.

La Log Loss (Entropie Croisée Binaire)

La MSE avec Sigmoïde crée une fonction non-convexe.

Nous utilisons la **Log Loss** :

$$J = -\frac{1}{m} \sum [y \log(\hat{p}) + (1 - y) \log(1 - \hat{p})]$$



Pénalité Sévère

Être confiant et avoir tort $\Rightarrow$ Coût $\rightarrow \infty$

Multi-Classe

Régression Softmax

Au-delà de la Classification Binaire

Et si nous avons K classes ? (ex : Chiffres MNIST 0-9)

Approches possibles :

- **Un-contre-Reste (One-vs-All)** : Entraîner K classifieurs binaires séparés.
- **Softmax** : Généraliser la Régression Logistique pour K classes directement.

Avantage de Softmax

Un seul modèle qui prédit les probabilités de toutes les classes simultanément.

Softmax : L'Idée

Étape 1 : Calculer un **score** (logit) pour chaque classe.

Chaque classe k a son propre vecteur de poids $\theta^{(k)}$:

$$s_k(x) = (\theta^{(k)})^T x$$

Problème : Les scores peuvent être négatifs ou très grands.

Étape 2 : Transformer les scores en **probabilités** (entre 0 et 1, somme = 1).

La Fonction Softmax

$$\hat{p}_k = \frac{e^{s_k(x)}}{\sum_{j=1}^{K} e^{s_j(x)}}$$

Pourquoi cette formule ?

- **Exponentielle (e^s)** : Transforme tout score en valeur positive.
- **Normalisation (diviser par la somme)** : Garantit que $\sum_k \hat{p}_k = 1$.
- **Amplification** : Les grands scores deviennent des probabilités proches de 1, les petits proches de 0.

Softmax : Exemple Concret

Supposons 3 classes avec les scores : $s = [2.0, 1.0, 0.1]$

Étape 1 : Exponentier chaque score

$$e^{2.0} = 7.39, \quad e^{1.0} = 2.72, \quad e^{0.1} = 1.11$$

Étape 2 : Calculer la somme

$$\text{Somme} = 7.39 + 2.72 + 1.11 = 11.22$$

Étape 3 : Normaliser

$$\hat{p} = \left[\frac{7.39}{11.22}, \frac{2.72}{11.22}, \frac{1.11}{11.22}\right] = [0.66, 0.24, 0.10]$$

$\Rightarrow$ La classe 1 a la plus haute probabilité (66%).

Encodage One-Hot

Représentation de la vraie classe :

Si la vraie classe est la classe 2 parmi 3 classes :

$$y = [0, 1, 0]$$

- La position de la vraie classe contient 1
- Toutes les autres positions contiennent 0

Entropie Croisée Catégorielle

$$\boxed{J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(\hat{p}_k^{(i)})}$$

Simplifié : Puisque $y_k = 0$ sauf pour la vraie classe c :

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m} \log(\hat{p}_c^{(i)})$$

Interprétation :

- Si $\hat{p}_{\text{vraie classe}} \approx 1 : -\log(1) = 0$ (coût faible).
- Si $\hat{p}_{\text{vraie classe}} \approx 0 : -\log(0) \to \infty$ (coût énorme).

Résumé du Chapitre 2 : Points Clés (1/2)

- **Régression Linéaire** utilise l'Équation Normale (petites données) ou la Descente de Gradient (grandes données).
- **Descente de Gradient** fonctionne en faisant des pas proportionnels au gradient négatif.
- **Taux d'Apprentissage** est critique : Trop petit = lent ; Trop grand = diverge.
- **SGD** est rapide et bruité ; **Batch** est lent et stable.

Résumé du Chapitre 2 : Points Clés (2/2)

- **Polynômes** permettent aux modèles linéaires de s'ajuster aux données courbes.
- **Régularisation** (L1/L2) empêche le surapprentissage en pénalisant les grands poids.
- **Régression Logistique** utilise la Sigmoïde pour produire des probabilités.
- **Softmax** gère les problèmes multi-classes.

Pensée Finale

Les modèles linéaires sont les modèles les plus **interprétables** en ML.

Message Clé

Si vous comprenez la Descente de Gradient et la Log Loss ici, vous comprenez la boucle d'entraînement d'un Réseau de Neurones.