Cours : Régression logistique

Classification binaire, sigmoïde et tuning du modèle

K. Kadri

Plan

Problème : prédire une classe binaire

- On veut prédire une variable $y \in \{0, 1\}$ (ex : *malade* / *non malade*).
- Une régression linéaire donnerait une valeur réelle non bornée.
- On a besoin d'un modèle qui donne une **probabilité** entre 0 et 1 :

$$p(y = 1 \mid \mathrm{x}) \in [0, 1]$$

Idée clé

La régression logistique ne prédit pas directement une classe, mais une **probabilité de classe positive**, puis applique un **seuil** (souvent 0,5).

La fonction sigmoïde

Définition

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

où $z = \mathrm{w}^\top \mathrm{x} + b$.

- Pour $z \to +\infty$: $\sigma(z) \to 1$.
- Pour $z \to -\infty$: $\sigma(z) \to 0$.
- Interprétation : $\sigma(z)$ est une **probabilité**.

Modèle logistique

$$p(y = 1 \mid \mathrm{x}) = \sigma(\mathrm{w}^\top \mathrm{x} + b)$$

Log-odds et modèle linéaire

- On peut réécrire le modèle sous forme de **log-odds** :

$$\log \frac{p}{1-p} = \mathrm{w}^\top \mathrm{x} + b$$

- $\dfrac{p}{1-p}$: **odds** = rapport de chances.
- L'algorithme apprend w et b qui expliquent ce rapport de chances.

Interprétation

Chaque coefficient w_j mesure l'effet de la feature x_j sur le **log-odds** d'appartenir à la classe positive.

Fonction de coût : log-loss

Pour un exemple (x_i, y_i)

$$\mathcal{L}(y_i, \hat{p}_i) = -\Big[y_i \log(\hat{p}_i) + (1 - y_i)\log(1 - \hat{p}_i)\Big]$$

où $\hat{p}_i = p(y = 1 \mid \mathrm{x}_i)$.

- Si le modèle est sûr et a raison $\rightarrow$ perte faible.
- Si le modèle est sûr et se trompe $\rightarrow$ perte très grande.

Objectif d'apprentissage

Minimiser la somme (ou moyenne) de la log-loss sur tout le jeu d'entraînement.

Régularisation L2 et L1

- **Problème** : sur-apprentissage (overfitting) si trop de liberté sur les coefficients.
- **Régularisation L2** (Ridge) :

$$\mathcal{L}_{\text{totale}} = \mathcal{L}_{\text{log-loss}} + \lambda \|w\|_2^2$$

- **Régularisation L1** (Lasso) :

$$\mathcal{L}_{\text{totale}} = \mathcal{L}_{\text{log-loss}} + \lambda \|w\|_1$$

Effet

- L2 : rétrécit tous les coefficients (mais rarement à 0).
- L1 : pousse certains coefficients exactement à 0 $\rightarrow$ **sélection de variables**.

Hyperparamètre C dans scikit-learn

Dans `LogisticRegression`

$$C = \frac{1}{\lambda}$$

- Petit $C \to$ grande régularisation (modèle plus simple, plus biaisé).
- Grand $C \to$ faible régularisation (modèle plus complexe, plus à risque d'overfitting).

Tuning

On choisit C (et le type de pénalité) par **validation croisée** (GridSearch, RandomizedSearch).

Seuil de décision

- Par défaut :

$$\hat{y} = \begin{cases} 1 & \text{si } p(y = 1 \mid \mathrm{x}) \geq 0{,}5 \\ 0 & \text{sinon} \end{cases}$$

- Mais on peut **adapter le seuil** en fonction du problème :
  - seuil plus bas $\rightarrow$ plus de positifs détectés (rappel $\uparrow$, risque FP $\uparrow$).
  - seuil plus haut $\rightarrow$ moins de faux positifs mais rappel $\downarrow$.

Lien avec les métriques

On choisit le seuil en fonction du **compromis** Précision / Rappel / F1-score et du **coût métier** des faux positifs et des faux négatifs.

Courbe ROC et AUC

- En faisant varier le seuil, on obtient plusieurs points $(\mathrm{FPR}, \mathrm{TPR})$.
- La courbe ROC trace :

$$\mathrm{TPR} \quad \text{en fonction de} \quad \mathrm{FPR}$$

- L'aire sous la courbe (AUC) mesure la **capacité globale** du modèle à séparer les classes.

Règle pratique

- AUC $\approx 0{,}5$: modèle aléatoire.
- AUC proche de 1 : très bonne séparation.

Avantages de la régression logistique

- Modèle **simple**, rapide à entraîner.
- Sortie **probabiliste** facile à interpréter.
- Interprétable : les coefficients peuvent être analysés.
- Fonctionne bien comme **baseline** sur beaucoup de problèmes.

Cas d'usage typiques

- Score de défaut de paiement (credit scoring).
- Probabilité de churn client.
- Diagnostic binaire (présence/absence d'une maladie).

Limites et bonnes pratiques

- Hypothèse de **frontière de décision linéaire** dans l'espace des features.
- Sensible aux **features mal échelles** $\rightarrow$ importance de standardiser.
- Peut être insuffisant si la relation est très non linéaire.

Bonnes pratiques

- Toujours standardiser les features continues.
- Tester plusieurs valeurs de C et types de pénalité.
- Examiner les coefficients et les métriques (F1, AUC) plutôt que la seule accuracy.