Université de Montpellier Année 2024-2025
HAX912X - Modèles linéaires généralisés

Examen de contrôle continu

Durée 1h - Aucun document autorisé
Vendredi 18 octobre 2024

Exercice 1 (6 pts) Décrire et commenter les résultats obtenus à l'aide du code Python ci-dessous. Aide paramètre `C` de la fonction `LogisticRegression` de `sklearn`

```
C : float, default=1.0
Inverse of regularization strength; must be a positive float.
Like in support vector machines, smaller values specify stronger regularization.
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_classification

X, y = make_classification(n_samples=300, n_features=40,
    n_classes=2,flip_y=0.1,class_sep=0.5, random_state=42)

X = StandardScaler().fit_transform(X)

model1 = LogisticRegression(penalty='l1', C=1, class_weight=None,
dual=False, solver = 'saga', fit_intercept=True, l1_ratio=None, max_iter=10000, tol=0.0001)
model1.fit(X,y)
print(model1.score(X,y))
0.8233333333333334

model2 = LogisticRegression(penalty='l1', C=0.1, class_weight=None,
dual=False, solver = 'saga', fit_intercept=True, l1_ratio=None, max_iter=10000, tol=0.0001)
model2.fit(X,y)
print(model2.score(X,y))
0.7866666666666666

from sklearn.model_selection import train_test_split
X,X_test,y,y_test = train_test_split(X,y,test_size=0.25)

model1 = LogisticRegression(penalty=None, class_weight=None,
dual=False, solver = 'saga', fit_intercept=True, l1_ratio=None, max_iter=10000, tol=0.0001)
model1.fit(X,y)
print(model1.score(X_test,y_test))
0.68

model2 = LogisticRegression(penalty='l1', C=0.1, class_weight=None,
dual=False, solver = 'saga', fit_intercept=True, l1_ratio=None, max_iter=10000, tol=0.0001)
model2.fit(X,y)
print(model2.score(X_test,y_test))
0.7466666666666667
```

Correction (comparaison des performances de modèles de régression logistique)

(3 pts) Dans la première partie du code, deux modèles de régression logistique sont appliqués à un jeu de données d'apprentissage comprenant $n = 300$ observations et $p = 40$ variables. Une méthode de régularisation Lasso est utilisée dans les deux cas. Pour le modèle 1, la régularisation avec $C = 1$ est moins stricte que pour le modèle 2, où $C = 0.1$. En effet, plus la valeur de C est petite, plus la régularisation est forte, ce qui tend à réduire les coefficients de régression, les rapprochant de zéro, voire à les annuler. Concernant les taux d'erreur de classification calculés sur l'échantillon d'apprentissage, le modèle 1 semble plus performant que le modèle 2. Cela est attendu, car nous sommes en situation de sur-apprentissage, où le modèle le plus complexe est favorisé.

(3 pts) Dans la deuxième partie du code, deux modèles de régression logistique sont appliqués à un jeu de données d'apprentissage comprenant $n = 225$ observations et $p = 40$ variables. Une méthode de régularisation Lasso est utilisée uniquement pour le deuxième modèle. Le modèle 1 n'est pas régularisé, tandis que pour le modèle 2, une régularisation avec $C = 1$ est appliquée. Un échantillon de test comprenant 75 individus est utilisé pour évaluer les performances des deux modèles. Le modèle régularisé montre les meilleures performances. Cette différence s'explique par le fait que nous ne sommes pas en situation de sur-apprentissage, grâce à l'utilisation d'un échantillon de test indépendant.

Exercice 2 (5 pts) Nous considérons n variables aléatoires indépendantes $y_1, \dots, y_n$ telles que y_i est distribuée suivant une loi binomiale de paramètre $(100, p_i)$. Pour tout $i = 1, \dots, n$, nous supposons que $\log(-\log(1 - p_i)) = x_i^T \beta$ où $x_i \in \mathbb{R}^p$ est un vecteur supposé connu et β un vecteur de paramètres inconnus. Montrer qu'il s'agit d'un modèle linéaire généralisé. La fonction de lien canonique a-t-elle été utilisée ?

Correction

$$f(y; p) = C_{100}^y p^y (1-p)^{100-y} \mathbf{1}_{\{0,\dots,100\}}(y) \delta(dy)$$
$$f(y; p) = \exp\left(y \log(p/(1-p)) + 100 \log(1-p)\right) \nu(dy)$$

La loi binomiale appartient à la famille exponentielle scalaire avec

$$\theta = \log(p/(1-p)) \quad \Longleftrightarrow \quad p = \exp(\theta)/(1+\exp(\theta))\,,$$

et

$$b(\theta) = 100 \log(1 + \exp(\theta))\,.$$

Par ailleurs,

$$\log(-\log(1 - \mathbb{E}(y|x)/100)) = x^T \beta\,,$$

Il s'agit donc bien d'un modèle linéaire généralisé.

Aussi,

$$x^T \beta = \log(-\log(1 - \exp(\theta)/(1+\exp(\theta)))) = \log(\log(1+\exp(\theta))) \neq \theta\,,$$

ce n'est donc pas le lien canonique qui a été utilisé.

Exercice 3 (4 pts) Décrire et commenter les résultats obtenus à l'aide du code Python ci-dessous.

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_regression
import numpy as np

X, y = make_regression(n_samples=100, n_features=80, noise=0.1, random_state=42)

scaler = StandardScaler()
X_normalized = scaler.fit_transform(X)
```

```
model1 = LinearRegression()
scores1 = cross_val_score(model1, X_normalized, y, cv=5, scoring='neg_mean_absolute_error')
print(scores1)
print(np.mean(scores1))

[-10.96963938 -44.27345903 -40.90757601 -32.031635   -18.96821433]
-29.430104749593802

alpha = 0.1
model2 = Lasso(alpha=alpha)
scores2 = cross_val_score(model2, X_normalized, y, cv=5, scoring='neg_mean_absolute_error')
print(scores2)
print(np.mean(scores2))

[-0.34950166 -0.3056262  -0.30595072 -0.46432015 -0.29677541]
-0.3444348268695149
```

Correction (comparaison des performances entre la régression linéaire et la régression Lasso)

(2 pts) Le modèle de régression linéaire produit des scores d'erreur absolue moyenne (MAE) calculés par validation croisée à 5 ensembles importants, avec une valeur moyenne d'environ 29,43. La dispersion de ces erreurs entre les ensembles est significative, certains ensembles affichant des performances bien plus mauvaises que d'autres, Cela indique que le modèle est en situation de sur-ajustement aux données, en raison du nombre élevé de variables explicatives $p = 80$ par rapport à la taille $n = 100$ de l'échantillon.

(2 pts) Le modèle Lasso, en revanche, donne de bien meilleurs résultats en termes de MAE. La MAE moyenne est significativement plus basse, autour de 0,34. Les scores de chaque ensemble sont plus cohérents, allant d'environ 0,30 à 0,46. Le Lasso introduit une régularisation (contrôlée par le paramètre alpha, fixé ici à $0,1$), qui permet d'atténuer le sur-ajustement en réduisant les valeurs les coefficients des variables les moins importantes. Il en résulte un modèle plus robuste. L'amélioration des performances avec Lasso démontre la puissance de la régularisation dans les ensembles de données en grande dimension (où le nombre de variables est important par rapport à la taille de l'échantillon).

Exercice 4 (5 pts)

1) (2,5 pts) Quelle est la différence entre les modèles logit et probit ?

2) (2,5 pts) Quelle est l'objectif des méthodes de régularisation en regression ?

Correction

1) Les modèles logit et probit sont deux modèles linéaires généralisées pour modéliser des variables à expliquer binaires. Ils sont très similaires, mais diffèrent principalement dans leur fonction de lien, qui relie la probabilité de l'événement à la combinaison linéaire des variables explicatives. Le logit utilise la fonction logistique, tandis que le probit utilise la distribution normale cumulative.

2) Les méthodes de régularisation en régression visent à éviter le surapprentissage (overfitting) en pénalisant les modèles complexes. Elles ajoutent une contrainte sur la taille des coefficients pour favoriser des modèles plus simples et plus généralisables. On introduit un biais mais on réduit la variance des estimations des coefficients de régression.