

1.SKLearn Lasso 回归进行特征筛选

一、Lasso 回归的原理

1.1 什么是 Lasso 回归

- **Lasso (Least Absolute Shrinkage and Selection Operator)** 回归是一种在线性回归基础上加入了 **L1 正则化** (L1 范数惩罚项) 的线性模型。
- **目的**: 通过对模型参数的绝对值之和进行惩罚, 实现参数的稀疏化, 从而达到 **特征选择** 的效果。

1.2 Lasso 回归的损失函数

Lasso 回归的损失函数如下:

$$\text{Loss} = \frac{1}{2n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^p |w_j|$$

- n : 样本数量
- y_i : 真实值
- \hat{y}_i : 预测值
- w_j : 模型的第 j 个参数
- α : 正则化强度 (惩罚项系数)

1.3 L1 正则化的作用

- **参数稀疏化**: L1 正则化会使一些模型参数 w_j 收缩为 零, 从而达到 **特征选择** 的效果。
- **防止过拟合**: 通过惩罚模型的复杂度, 避免模型在训练集上过拟合, 提升模型的泛化能力。

$$\text{Loss} = \frac{1}{2n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^p |w_j|$$

- n : 样本数量
- y_i : 真实值
- \hat{y}_i : 预测值
- w_j : 模型的第 j 个参数
- α : 正则化强度 (惩罚项系数)

1.3 L1 正则化的作用

- **参数稀疏化**: L1 正则化会使一些模型参数 w_j 收缩为 **零**, 从而达到 **特征选择** 的效果。
- **防止过拟合**: 通过惩罚模型的复杂度, 避免模型在训练集上过拟合, 提升模型的泛化能力。

二、LassoCV 的原理

2.1 交叉验证选择最佳 α

- **问题**: 在实际应用中, 很难预先知道哪个 α 值能够带来最佳的模型性能。
- **解决方案**: LassoCV 通过 **交叉验证 (Cross-Validation)**, 在一系列候选 α 值中自动选择使模型在验证集上表现最好的那个。

2.2 工作流程


1. **候选 α 值的生成**:
 - 生成一个 α 值的列表, 通常在对数尺度上均匀分布。
2. **交叉验证**:
 - 对于每个候选 α 值, 进行 K 折交叉验证。
 - 在每一折中, 使用训练集训练模型, 使用验证集评估模型性能 (如均方误差)。
3. **选择最佳 α** :
 - 对每个 α , 计算所有折的平均验证误差。
 - 选择使平均验证误差最小的 α , 作为最佳正则化参数。

四、使用示例

4.1 导入库和数据

```
python

import numpy as np
from sklearn.linear_model import LassoCV
from sklearn.datasets import load_boston
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
```

 Copy code

4.2 加载数据

```
python

# 加载示例数据集（波士顿房价数据）
X, y = load_boston(return_X_y=True)
```

 Copy code

4.3 建立 LassoCV 模型

```
python

# 创建一个包含标准化和 LassoCV 的管道
model = make_pipeline(
    StandardScaler(),
    LassoCV(cv=5, random_state=42)
)

# 拟合模型
model.fit(X, y)
```

 Copy code

4.4 查看结果

python

Copy code

```
# 获取最佳的 alpha 值
lasso_cv = model.named_steps['lassocv']
best_alpha = lasso_cv.alpha_
print(f"Best alpha selected by cross-validation: {best_alpha}")

# 获取模型的系数
coefficients = lasso_cv.coef_
print("Coefficients:")
print(coefficients)

# 查看被选择的特征（系数非零）
selected_features = np.where(coefficients != 0)[0]
print("Selected feature indices:")
print(selected_features)
```

五、注意事项

5.1 特征标准化

- **必要性：**Lasso 回归对特征的尺度敏感，特征值的大小会影响惩罚项的作用。
- **解决方法：**在拟合之前对特征进行标准化（均值为 0，方差为 1），以确保所有特征在同一尺度上。
- **实现方式：**使用 `StandardScaler`，或者在 `Pipeline` 中添加标准化步骤。

5.2 处理 `normalize` 参数弃用

- 在 `scikit-learn` 0.24 版本之后，`normalize` 参数已被弃用。
- **建议：**使用 `StandardScaler` 或 `Pipeline` 对数据进行标准化。

5.3 选择合适的 `cv` 参数

- **默认值：**`cv=None`，即使用 5 折交叉验证。
- **自定义：**可以根据数据集的大小和特性，指定适当的折数，或者使用特定的交叉验证策略。

5.4 解释模型结果

- **系数为零的特征：**被认为对模型贡献较小，被 Lasso 回归剔除。
- **系数非零的特征：**被认为对预测目标有显著贡献。

`StandardScaler` 是 `scikit-learn` 库中的一个工具，用于对特征数据进行标准化处理。它按照特征（即列）对数据进行均值归一化和方差缩放，使得每个特征的数据分布具有零均值和单位方差。

六、总结

- **LassoCV** 通过在一系列 **alpha** 值上进行交叉验证，自动选择最佳的正则化参数。
- **参数解释**：了解各参数的含义，有助于根据数据和任务需求调整模型，提高性能。
- **特征选择**：Lasso 回归能够实现特征选择，对于高维数据尤为有用。
- **标准化的重要性**：特征标准化能够确保模型收敛和结果的可靠性。

2.1 权重的符号（正负）

- **正权重** ($w_j > 0$) :
 - 特征 x_j 与目标变量 y **正相关**。
 - 当 x_j 增加, y 也倾向于增加。
 - 对预测有积极的推动作用。
- **负权重** ($w_j < 0$) :
 - 特征 x_j 与目标变量 y **负相关**。
 - 当 x_j 增加, y 倾向于减少。
 - 对预测有抑制或减少的作用。

2.2 权重的绝对值大小

- **绝对值越大**（无论正负）：
 - 特征对预测目标的影响越大，重要性越高。
 - 说明特征的变化会显著影响预测结果。
- **绝对值越小**（接近零）：
 - 特征对预测目标的影响较小，重要性较低。
 - Lasso 回归可能会将权重压缩为零，表示特征被剔除。