Bài 4: HUẤN LUYỆN MÔ HÌNH VÀ LỖI

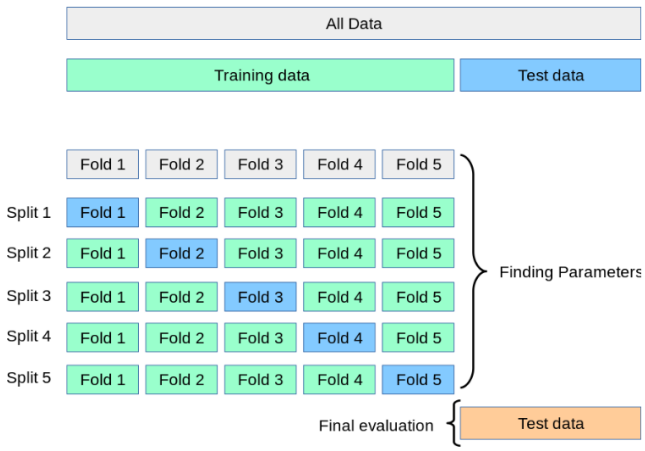
Hướng dẫn 4.1: k-fold cross-validation với mô hình Hồi quy

1. Quy trình xây dựng k-fold cross-validation

Sau khi chia tập dữ liệu *D* ban đầu thành 2 tập train – test với tỉ lệ thông thường là 70% - 30%, quá trình huấn luyện diễn ra với tập D_{train} được tiến hành bằng cách lấy ra từ D_{train} một phần nhỏ dữ liệu làm *tập dữ liệu kiểm thử cho bước huấn luyện thứ i – gọi là* $D_{validation}$. Phần dữ liệu còn lại đóng vai trò là tập dữ liệu huấn luyện. Quá trình này lặp lại nhiều lần cho đến khi mô hình huấn luyện đạt được yêu cầu. Tóm lại, quy trình thực hiện *k-fold cross-validation* (với k thông thường được chọn k = 10 – nên gọi là *10-fold cross-validation*) được tóm tắt như sau:

- Bước 1: Đọc dữ liệu gốc *D*
- Bước 2: Điều chỉnh dữ liệu
- Bước 3: Chia train – test theo tỉ lệ (thông thường là 70% - 30%)
- Bước 4: Xác định k
- Bước 5: Thực hiện huấn luyên mô hình với k-fold cross validation
- Bước 6: Kiểm định mô hình với tập dữ liệu test

Minh họa quy trình này với 5-fold cross-validation theo ví dụ của sklearn



Hình 1: Minh họa quy trình 5-fold cross-validation (nguồn: sklearn)

2. Huấn luyện mô hình HQTT với 10-fold CV sử dụng Gradient Descent

```
import numpy as np
import os
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler

def computeLoss(X, y, w):
    m = y.shape[0]
    J = 0
    h = np.dot(X, w)
    J = (1/(2*m))*np.sum(np.square(h - y))
    return J

def gradientDescent(X, y, w, alpha, n):
    m = y.shape[0]
    J_history = []
    w_optimal = w.copy()
    for i in range(n):
        h = np.dot(X, w_optimal)
        error = h - y
        w_optimal = w_optimal - (alpha/m)*np.dot(X.T, error)
        J_history.append(computeLoss(X=X, y=y, w=w_optimal))
    return w_optimal, J_history

def main():
    #Bước 1: Đọc dữ liệu
    D = np.loadtxt(os.path.join('D:/data/hocmay', 'ex1data2.txt'),
delimiter=',')
    #Bước 2: Điều chỉnh dữ liệu - do đây là mô hình HQTT nên chấp nhận scale
cả vector y
    scaler = MinMaxScaler()
    scaler.fit(D)
    D = scaler.transform(D)
    #Bước 3: Phân chia train - test theo tỉ lệ 70% - 30%
    X, y = D[:, :-1], D[:, -1]
    x0 = np.ones((X.shape[0], 1))
    X = np.column_stack([x0, X])
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
random_state=15)
    #Bước 4: Xác định k-fold
    kf = KFold(n_splits=10)
    #Bước 5: Huấn luyện mô hình
    print('Huấn luyện mô hình với k-fold')
    #w opt = np.zeros((X.shape[1], 1))
    n = 1500
    alpha = 0.01
    step = 0
    for train2_index, val_index in kf.split(X = X_train,y = y_train):
        step = step + 1
        print('\tBước lặp huấn luyện thứ: ', step)
        w_opt = np.zeros((X.shape[1], 1))
        X_train2, X_val = X_train[train2_index], X_train[val_index]
        y_train2, y_val = y_train[train2_index], y_train[val_index]
        #Do chương trình chúng ta viết quy định y la mx1 nên cần reshape
        w_opt, J_history = gradientDescent(X=X_train2,
```

```
y=y_train2.reshape((y_train2.shape[0],1)),
                                        w=w_opt, alpha=alpha, n=n)
        print('\t\tĐánh giá mô hình trên tập dữ liệu validation')
        y_hat = np.dot(X_val, w_opt)
        print('\t\t\tMSE: ', mean_squared_error(y_val, y_hat))
    #Bước 6: Kiểm định mô hình với tập dữ liệu test
    print('ĐÁNH GIÁ HIỆU NĂNG CỦA MÔ HÌNH TRÊN TẬP DỮ LIỆU TEST')
    y_hat = np.dot(X_test, w_opt)
    print('\tMSE: ', mean_squared_error(y_test, y_hat))

if __name__ == '__main__':
    main()
```

3. Huấn luyện mô hình HQTT với 10-fold CV sử dụng LinearRegression

```
import numpy as np
import os
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression

#Bước 1: Đọc dữ liệu
D = np.loadtxt(os.path.join('D:/data/hocmay', 'ex1data2.txt'), delimiter=',')
#Bước 2: Điều chỉnh dữ liệu - do đây là mô hình HQTT nên chấp nhận scale cả
vector y
scaler = MinMaxScaler()
scaler.fit(D)
D = scaler.transform(D)
#Bước 3: Phân chia train - test theo tỉ lệ 70% - 30%
X, y = D[:, :-1], D[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
random_state=15)
#Bước 4: Xác định k-fold
kf = KFold(n_splits=10)
#Bước 5: Huấn luyện mô hình
print('Huấn luyện mô hình LinearRegression với k-fold')
model = LinearRegression()
step = 0
for train2_index, val_index in kf.split(X = X_train,y = y_train):
    step = step + 1
    print('\tBước lặp huấn luyện thứ: ', step)
    w_opt = np.zeros((X.shape[1], 1))
    X_train2, X_val = X_train[train2_index], X_train[val_index]
    y_train2, y_val = y_train[train2_index], y_train[val_index]
    model.fit(X_train2, y_train2)
    print('\t\tĐánh giá mô hình trên tập dữ liệu validation')
    y_hat = model.predict(X_val)
    print('\t\t\tMSE: ', mean_squared_error(y_val, y_hat))
#Bước 6: Kiểm định mô hình với tập dữ liệu test
print('ĐÁNH GIÁ HIỆU NĂNG CỦA MÔ HÌNH TRÊN TẬP DỮ LIỆU TEST')
y_hat = model.predict(X_test)
print('\tMSE: ', mean_squared_error(y_test, y_hat))
```

4. Câu hỏi mở rộng

1. Hãy viết lại chương trình main() của 2 ví dụ trên bằng cách phân rã các bước thành các hàm. Hàm do bạn tự đặt tên.
2. Hãy áp dụng k-fold cross-validation với các mô hình hồi quy khác như: ***Lasso*** hay ***Ridge***. Gửi mã lệnh chương trình vào phần trả lời bài tập