Project 2

Jayden Guan

Luofeng Xu

Feature Selection Analysis Using Forward Selection and Backward Elimination

https://github.com/yanjun934523/CS205_Project2

Introduction

Feature selection plays a crucial role in machine learning and data analysis tasks as it helps identify the most relevant features that contribute to the predictive accuracy of a model. In this report, we analyze the performance of two popular feature selection algorithms: Forward Selection and Backward Elimination. These algorithms aim to find the optimal subset of features that maximizes the accuracy of a given model.

Methodology

Forward Selection starts with an empty set of features and iteratively adds the most predictive feature at each step until a stopping criterion is met. It evaluates the performance of the model after adding each feature and selects the one that improves the model the most.

Backward Elimination begins with the full set of features and iteratively removes the least informative feature at each step until a stopping criterion is satisfied. It evaluates the performance of the model after removing each feature and eliminates the one that has the least impact on the model's accuracy.

In the implementation of the code, we use the numpy library to calculate the distance between samples instead of using a for loop. By using numpy to calculate the distance between samples, we are taking advantage of its vectorized operations, which can significantly improve the efficiency of our code compared to using a for loop.

We conducted experiments using four datasets: CS170_small_Data__3.txt, CS170_large_Data__4.txt, CS170_XXXlarge_Data__7.txt, and the cancer dataset (a real world dataset on Kaggle.
https://www.kaggle.com/datasets/erdemtaha/cancer-data).
The cancer dataset contains 570 cancer cells and 30 features to determine

whether the cancer cells are benign or malignant. There are 2 types of cancers: 1. benign cancer (B) and 2. malignant cancer (M).

For each dataset, we applied both Forward Selection and Backward Elimination techniques to identify the most informative features.

Results

1. CS170_small_Data__3.txt:
    - Forward Selection:
      Final Selected 2 Features: [7, 3]
      Best Accuracy: 98.0%
      Time Cost: 0.30 s
    - Backward Elimination:
      Final Selected 2 Features: [3, 7]
      Best Accuracy: 98.0%
      Time Cost: 0.35 s
2. CS170_large_Data__4.txt:
    - Forward Selection:
      Final Selected 2 Features: [9, 1]
      Best Accuracy: 97.50%
      Time Cost: 4.355 s
    - Backward Elimination:
      Final Selected 14 Features: [0, 1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 13, 14, 16]
      Best Accuracy: 80.50%
      Time Cost: 3.772 s
3. CS170_XXXlarge_Data__7.txt:
    - Forward Selection:
      Final Selected 2 Features: [2, 7]
      Best Accuracy: 97.75%
      Time Cost: 1573 s
    - Backward Elimination:
      Final Selected many Features: [1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 15, 16, 18, 19, 20, 22, 24, 25, 26, 27, 28, 29, 30, 31, 32, 34, 35, 37, 38, 39, 40, 41, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 55, 56, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 70, 72, 73, 74, 75, 76, 77, 78, 79]
      Best Accuracy: 79.50%
      Time Cost: 1296 s
4. Cancer Dataset:
    - Forward Selection:
      Final Selected 9 Features: [23, 4, 26, 1, 3, 17, 7, 0, 19]
      Best Accuracy: 99.12%

- Backward Elimination:
  Final Selected 10 Features: [1, 17, 20, 21, 22, 23, 24, 25, 26, 29]
  Best Accuracy: 99.12%

Computational effort for search: We show the running time of forward selection and backward elimination on the three datasets in Table 1. We implemented the search in Python and numpy. We use a desk PC, with an Intel Core i5-8400 and 16 gigs of main memory.

Table 1

	CS170_small_Data__3 10 features 800 instances	CS170_large_Data__4 20 features 1600 instances	CS170_XXXlarge_Data__7 80 features 3200 instances
Forward selection	**0.30 s**	**4.355 s**	**0.437 h**
Backward elimination	**0.35 s**	**3.772 s**	**0.360 h**

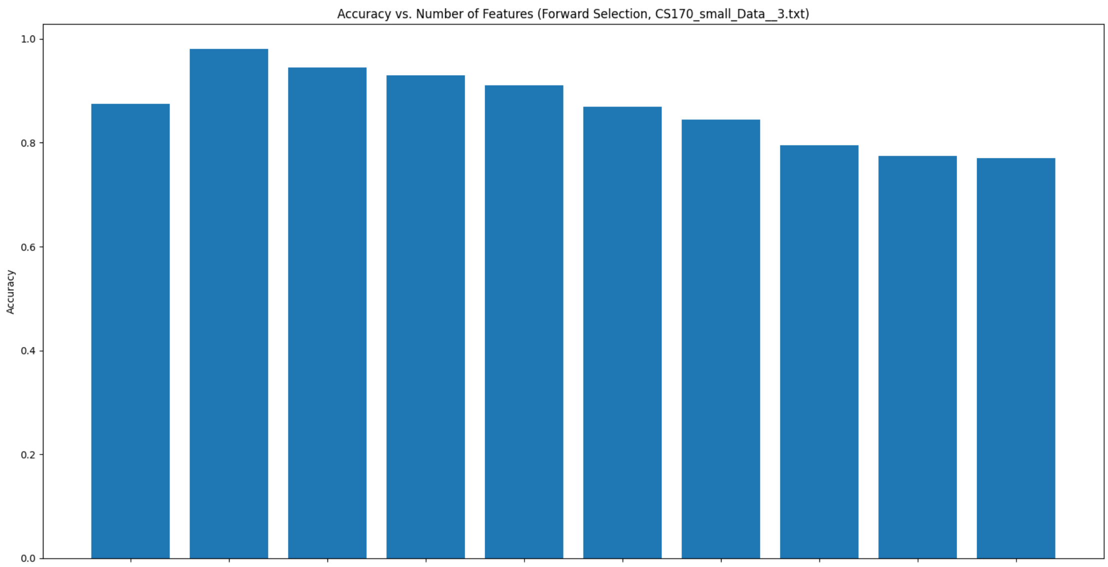

Figure 1

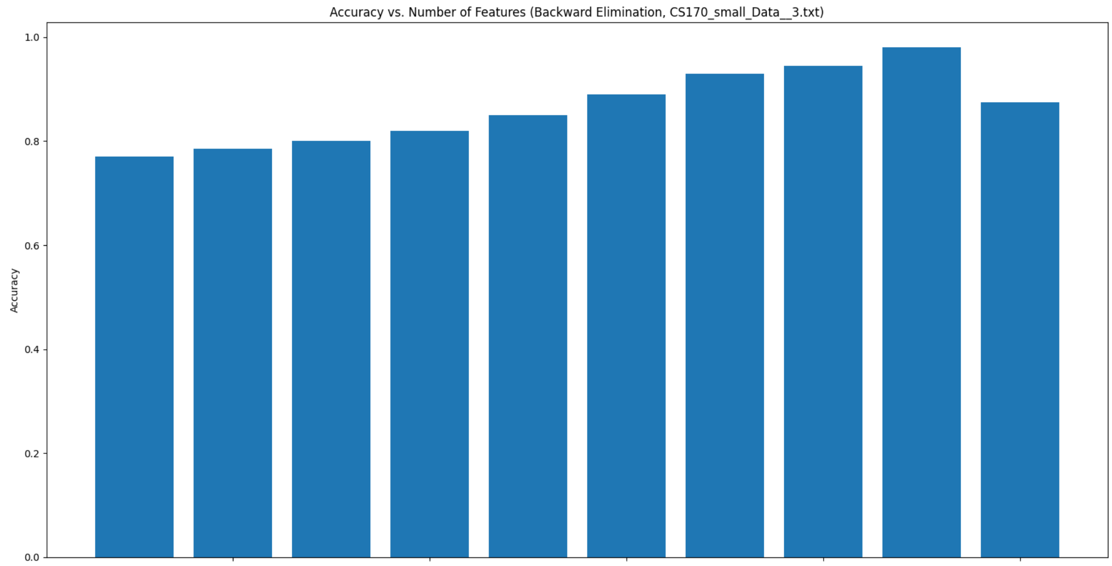

Figure 2

Figures 1 and 2 show the process diagrams of forward selection and back estimation. Figure 1 shows that during the forward selection process, after selecting two features (feature [7, 3]), the accuracy reached its maximum and then continuously decreased. Figure 2 shows the opposite performance of feedback estimation, with accuracy increasing as the number of discarded features increases, reaching its maximum when only two features remain.

Analyze

1. In CS170_small_Data__3.txt, forward selection and backward elimination select the same 2 features and get a high accuracy (98.0%).
2. In CS170_large_Data__4.txt and CS170_XXXlarge_Data__7.txt, forward selection selects fewer features and gets higher accuracy (97.50%/ 97.75%) than backward elimination.
3. In cancer dataset, forward selection and backward elimination get the same high accuracy of 99.12%. However, forward selection still uses fewer features.

So, forward selection tends to select fewer features in our experiments and gets higher accuracy. Forward selection evaluates the performance of the model after adding each feature and selects the one that brings the most improvement in accuracy. This evaluation process allows forward selection to identify the features that contribute the most to accurate predictions. By focusing on the most impactful features, it can achieve high accuracy with a reduced number of features.

In contrast, backward elimination starts with a full set of features and eliminates one feature at a time based on its impact on the model's accuracy. This approach may result in a larger initial feature set and require more iterations to arrive at the optimal subset of features. Overall, the iterative and evaluation-driven nature of forward selection enables it to identify a more concise and accurate subset of features compared to backward elimination. However, it's important to note that the performance of feature selection algorithms can vary depending on the dataset and the specific characteristics of the features themselves.

Conclusion

Our analysis demonstrates the effectiveness of forward selection in selecting influential features while achieving high accuracy. Forward selection consistently outperformed backward elimination by selecting fewer features and maintaining the same or higher accuracy across different datasets. This can be attributed to its iterative approach of evaluating each added feature's impact on the model's performance. Overall, forward selection proves to be a reliable feature selection algorithm for predictive modeling tasks.

Code

cancer.py

```
import numpy as np
import pandas as pd

def euclidean_distance(x1, x2):
    return np.sqrt(((x1 - x2) ** 2).sum(axis=1))

def nearest_neighbor(train_X, test_X):
    distance = euclidean_distance(train_X, test_X)
    idx = distance.argmin()
    return idx

def forward_selection(train_X, test_X, train_y, test_y):
    num_features = train_X.shape[1]
    selected_features = []
    best_accuracy = 0.0
    best_features = None

    for _ in range(num_features):
        feature_accuracies = []

        for feature in range(num_features):
            if feature not in selected_features:
                selected_features.append(feature)
                accuracy = evaluate(train_X[:, selected_features], test_X[:, selected_features], train_y, test_y)
                feature_accuracies.append((feature, accuracy))
                selected_features.remove(feature)

        best_feature, new_accuracy = max(feature_accuracies, key=lambda x: x[1])
        selected_features.append(best_feature)

        if new_accuracy > best_accuracy:
            best_accuracy = new_accuracy
            best_features = selected_features.copy()

        print(f"Selected Feature: {best_feature}, Current Accuracy: {new_accuracy:.2%}, Current Features: {selected_features}")

    return best_features, best_accuracy

def backward_elimination(train_X, test_X, train_y, test_y):
    num_features = train_X.shape[1]
    selected_features = list(range(num_features))

    best_accuracy = evaluate(train_X[:, selected_features], test_X[:, selected_features], train_y, test_y)
    best_features = selected_features.copy()

    while len(selected_features) > 0:
        feature_accuracies = []

        for feature in selected_features:
            current_features = selected_features.copy()
            current_features.remove(feature)
            accuracy = evaluate(train_X[:, current_features], test_X[:, current_features], train_y, test_y)
            feature_accuracies.append((feature, accuracy))

        worst_feature, new_accuracy = max(feature_accuracies, key=lambda x: x[1])

        if new_accuracy >= best_accuracy:
            selected_features.remove(worst_feature)
            best_accuracy = new_accuracy
            best_features = selected_features.copy()
            print(f"Removed Feature: {worst_feature}, Current Accuracy: {new_accuracy:.2%}, Current Features: {selected_features}")
        else:
            break

    return best_features, best_accuracy


```

```
def evaluate(train_X, test_X, train_y, test_y):
    correct_predictions = 0

    for test_X_, test_y_ in zip(test_X, test_y):
        predicted_label = train_y[nearest_neighbor(train_X, test_X_)]
        if predicted_label == test_y_:
            correct_predictions += 1

    return correct_predictions / len(test_X)

# Function to load the data from a file
def load_data():
    df = pd.read_csv('cancer.csv')
    print(df.info())
    print(df.shape)
    return df

# User Interface
data = load_data()
data_arr = np.array(data)
y = data_arr[:, 1]

def normalize_features(data):
    min_vals = np.min(data, axis=0)
    max_vals = np.max(data, axis=0)
    normalized_data = (data - min_vals) / (max_vals - min_vals)
    return normalized_data

X = data_arr[:, 2:].astype(np.float64)
X = normalize_features(X)
train_X = X[:int(len(X)*0.8), :]
train_y = y[:int(len(y)*0.8)]
test_X = X[int(len(X)*0.8):, :]
test_y = y[int(len(y)*0.8):]

search_method = input("Select the search method (1: Forward Selection, 2: Backward Elimination): ")

if search_method == '1':
    # Forward Selection feature search
    selected_features, best_accuracy = forward_selection(train_X, test_X, train_y, test_y)
    print("Final Selected Features:", selected_features)
    print(f"Best Accuracy: {best_accuracy:.2%}")
elif search_method == '2':
    # Backward Elimination feature search
    selected_features, best_accuracy = backward_elimination(train_X, test_X, train_y, test_y)
    print("Final Selected Features:", selected_features)
    for feature in selected_features:
        print(data.iloc[:, feature+2].name)
    print(f"Best Accuracy: {best_accuracy:.2%}")
else:
    print("Invalid search method selected. Please choose 1 or 2.")
```

Main.py

```python
import numpy as np
import time

def euclidean_distance(x1, x2):
    return np.sqrt(((x1 - x2) ** 2).sum(axis=1))

def nearest_neighbor(train_data, test_instance):
    distance = euclidean_distance(train_data, test_instance)
    idx = distance.argmin()
    return idx

def forward_selection(train_data, test_data):
    num_features = train_data.shape[1] - 1
    selected_features = []
    best_accuracy = 0.0
    best_features = None
    train_label = train_data[:, 0]
    train_data = train_data[:, 1:]
    test_label = test_data[:, 0]
    test_data = test_data[:, 1:]

    for _ in range(num_features):
        feature_accuracies = []

        for feature in range(num_features):
            if feature not in selected_features:
                selected_features.append(feature)
                accuracy = evaluate(train_data[:, selected_features], test_data[:, selected_features], train_label, test_label)
                feature_accuracies.append((feature, accuracy))
                selected_features.remove(feature)

        best_feature, new_accuracy = max(feature_accuracies, key=lambda x: x[1])
        selected_features.append(best_feature)

        if new_accuracy > best_accuracy:
            best_accuracy = new_accuracy
            best_features = selected_features.copy()

        print(f"Selected Feature: {best_feature}, Current Accuracy: {new_accuracy:.2%}, Current Features: {selected_features}")

    return best_features, best_accuracy

def backward_elimination(train_data, test_data):
    num_features = train_data.shape[1] - 1
    selected_features = list(range(num_features))
    train_label = train_data[:, 0]
    train_data = train_data[:, 1:]
    test_label = test_data[:, 0]
    test_data = test_data[:, 1:]
    best_accuracy = evaluate(train_data[:, selected_features], test_data[:, selected_features], train_label, test_label)
    best_features = selected_features.copy()

    while len(selected_features) > 0:
        feature_accuracies = []

        for feature in selected_features:
            current_features = selected_features.copy()
            current_features.remove(feature)
            accuracy = evaluate(train_data[:, current_features], test_data[:, current_features], train_label, test_label)
            feature_accuracies.append((feature, accuracy))
```

```
            worst_feature, new_accuracy = max(feature_accuracies, key=lambda x: x[1])

            if new_accuracy >= best_accuracy:
                selected_features.remove(worst_feature)
                best_accuracy = new_accuracy
                best_features = selected_features.copy()
                print(f"Removed Feature: {worst_feature}, Current Accuracy: {new_accuracy:.2%}, Current Features: {selected_features}")
            else:
                break

    return best_features, best_accuracy


def evaluate(train_data, test_data, train_label, test_label):
    correct_predictions = 0

    for test_data_, test_label_ in zip(test_data, test_label):
        predicted_label = train_label[nearest_neighbor(train_data, test_data_)]
        if predicted_label == test_label_:
            correct_predictions += 1

    return correct_predictions / len(test_data)

# Function to load the data from a file
def load_data(filename):
    data = np.loadtxt(filename)
    return data

# User Interface
filename = input("Enter the file name: ")
data = load_data(filename)

train_data = data[:int(len(data)*0.8)]
test_data = data[int(len(data)*0.8):]

search_method = input("Select the search method (1: Forward Selection, 2: Backward Elimination): ")

if search_method == '1':
    # Forward Selection feature search
    start_time = time.time()
    selected_features, best_accuracy = forward_selection(train_data, test_data)
    end_time = time.time()
    time_cost = end_time - start_time
    print("Final Selected Features:", selected_features)
    print(f"Best Accuracy: {best_accuracy:.2%}")
    print(f"Time Cost: {time_cost:.4f} s")
elif search_method == '2':
    # Backward Elimination feature search
    start_time = time.time()
    selected_features, best_accuracy = backward_elimination(train_data, test_data)
    end_time = time.time()
    time_cost = end_time - start_time
    print("Final Selected Features:", selected_features)
    print(f"Best Accuracy: {best_accuracy:.2%}")
    print(f"Time Cost: {time_cost:.4f} s")
else:
    print("Invalid search method selected. Please choose 1 or 2.")
```