prac2

October 18, 2024

Aim: Implementing linear classifier using Linear discriminant function

Theory: ### Implementing a Linear Classifier Using Linear Discriminant Function (LDF)

A **Linear Discriminant Function (LDF)** is a method for classifying data by finding a linear boundary between different classes. Here's how you can approach building such a classifier without getting into mathematical equations.

0.0.1 1. Understanding the Linear Discriminant Function

The core idea behind a linear classifier using LDF is that it assigns data points to different classes based on their features. It does this by defining a line (or plane in higher dimensions) that best separates the data into distinct groups. The decision of which class a data point belongs to depends on which side of the line (or plane) it falls on.

0.0.2 2. Setting Up the Classifier

- **Input Features**: The model takes an input consisting of several features (e.g., height, weight, age) that describe each data point.
- **Weights and Bias**: The classifier uses weights (which are like importance factors) for each feature and a bias (similar to an offset) to decide the classification.
- **Score Calculation**: For each input, the classifier computes a score by combining the features with the weights and bias. This score is then used to decide the class.

0.0.3 3. Classification Decision

- **Positive Class**: If the score is above a certain threshold, the data point is classified into one group (e.g., "Class 1").
- **Negative Class**: If the score is below or equal to that threshold, it is classified into the other group (e.g., "Class 2").

0.0.4 4. Training the Classifier

During training, the classifier adjusts the weights and bias based on the training data. The goal is to tweak these values so that the decision boundary (the line or plane separating the classes) correctly classifies as many data points as possible. In some methods, this is done by minimizing the error made on the training data.

0.0.5 5. Linear Decision Boundary

Since the function is linear, the boundary between the classes is a straight line (or a plane in higher dimensions). For example, in two dimensions, the classifier separates the points using a line, and in three dimensions, it uses a plane. This boundary helps decide the classification of new data points.

0.0.6 6. Application

Once the classifier is trained, you can use it to classify new, unseen data. You feed the input features into the classifier, and it will output a class label based on the learned boundary.

0.0.7 7. Usage in Practice

- **Binary Classification**: LDF is commonly used for binary classification, where you want to separate data into two categories, such as "spam" and "not spam."
- **Extension to Multiple Classes**: With some extensions, LDF can also handle more than two classes by creating multiple boundaries to separate each class.

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import accuracy_score, confusion_matrix,␣
↪classification_report
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
df = pd.read_csv('heart.csv')
```

[3]:
```
df.head(5)
```

[3]:
```
   age  sex  cp  trtbps  chol  fbs  restecg  thalachh  exng  oldpeak  slp  \
0   63    1   3     145   233    1        0       150     0      2.3    0
1   37    1   2     130   250    0        1       187     0      3.5    0
2   41    0   1     130   204    0        0       172     0      1.4    2
3   56    1   1     120   236    0        1       178     0      0.8    2
4   57    0   0     120   354    0        1       163     1      0.6    2

   caa  thall  output
0    0      1       1
1    0      2       1
2    0      2       1
3    0      2       1
4    0      2       1
```

[4]:
```
df.describe()
```

[4]:

```
              age         sex          cp      trtbps        chol         fbs  \
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean    54.366337    0.683168    0.966997  131.623762  246.264026    0.148515
std      9.082101    0.466011    1.032052   17.538143   51.830751    0.356198
min     29.000000    0.000000    0.000000   94.000000  126.000000    0.000000
25%     47.500000    0.000000    0.000000  120.000000  211.000000    0.000000
50%     55.000000    1.000000    1.000000  130.000000  240.000000    0.000000
75%     61.000000    1.000000    2.000000  140.000000  274.500000    0.000000
max     77.000000    1.000000    3.000000  200.000000  564.000000    1.000000

          restecg    thalachh        exng     oldpeak         slp         caa  \
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean     0.528053  149.646865    0.326733    1.039604    1.399340    0.729373
std      0.525860   22.905161    0.469794    1.161075    0.616226    1.022606
min      0.000000   71.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000  133.500000    0.000000    0.000000    1.000000    0.000000
50%      1.000000  153.000000    0.000000    0.800000    1.000000    0.000000
75%      1.000000  166.000000    1.000000    1.600000    2.000000    1.000000
max      2.000000  202.000000    1.000000    6.200000    2.000000    4.000000

            thall      output
count  303.000000  303.000000
mean     2.313531    0.544554
std      0.612277    0.498835
min      0.000000    0.000000
25%      2.000000    0.000000
50%      2.000000    1.000000
75%      3.000000    1.000000
max      3.000000    1.000000
```

[5]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trtbps    303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalachh  303 non-null    int64
 8   exng      303 non-null    int64
 9   oldpeak   303 non-null    float64
```

```
 10  slp       303 non-null    int64
 11  caa       303 non-null    int64
 12  thall     303 non-null    int64
 13  output    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

[6]:
```
df.shape
```

[6]:
```
(303, 14)
```

[7]:
```
X = df.drop('output', axis=1)
y = df['output']
```

[8]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)
```

[9]:
```
X_train.head()
```

[9]:
```
     age  sex  cp  trtbps  chol  fbs  restecg  thalachh  exng  oldpeak  slp  \
124   39    0   2      94   199    0        1       179     0      0.0    2
72    29    1   1     130   204    0        0       202     0      0.0    2
15    50    0   2     120   219    0        1       158     0      1.6    1
10    54    1   0     140   239    0        1       160     0      1.2    2
163   38    1   2     138   175    0        1       173     0      0.0    2

     caa  thall
124    0      2
72     0      2
15     0      2
10     0      2
163    4      2
```

[10]:
```
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[11]:
```
X_train = pd.DataFrame(X_train, columns=X.columns)
X_test = pd.DataFrame(X_test, columns=X.columns)
```

[12]:
```
lda = LinearDiscriminantAnalysis()

param_grid = {
    'solver': ['svd', 'lsqr', 'eigen'],
    'shrinkage': [None, 'auto', 0.5]
}
```

```python
grid_search = GridSearchCV(estimator=lda, param_grid=param_grid, cv=5,␣
↪n_jobs=-1, scoring='accuracy')

grid_search.fit(X_train, y_train)

print("Best Parameters:", grid_search.best_params_)

best_lda = grid_search.best_estimator_
y_pred = best_lda.predict(X_test)

print("Classification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:\n", accuracy_score(y_test, y_pred))
```

```
Best Parameters: {'shrinkage': 0.5, 'solver': 'lsqr'}
Classification Report:
               precision    recall  f1-score   support

           0       0.77      0.83      0.80        41
           1       0.85      0.80      0.82        50

    accuracy                           0.81        91
   macro avg       0.81      0.81      0.81        91
weighted avg       0.82      0.81      0.81        91

Accuracy Score:
 0.8131868131868132
```

0.0.8 Conclusion

A linear classifier using a linear discriminant function is a straightforward method for classifying data, especially when the relationship between the classes is linear. It works by assigning weights to features, calculating a score, and then using that score to classify data into different groups. The simplicity of this approach makes it effective for many real-world applications where a linear decision boundary is sufficient.