Лабораторная работа 10
Наивный Байесовский классификатор

Гузовская Александра Чеславовна
Б9123-01.03.02сп

9 июня 2025 г.

Матчасть

Наивный Байесовский классификатор основан на теореме Байеса со строгим (наивным) предположением о независимости признаков между собой при заданном классе.
Для объекта с признаками $x = (x_1, x_2, ..., x_m)$ вероятность принадлежности к классу y_k вычисляется как:

$$P(y_k|x) = P(y_k) \cdot \prod \frac{P(x_i|y_k)}{P(x)}$$

где:
> $P(y_k)$ — априорная вероятность класса y_k
> $P(x_i|y_k)$ — вероятность признака x_i для класса y_k
> $P(x)$ — нормирующая константа

Для дискретных признаков $P(x_i|y_k)$ оценивается относительной частотой с добавлением сглаживания Лапласа:

$$P(x_i|y_k) = \frac{(N(x_i, y_k) + \alpha)}{(N(y_k) + \alpha \cdot n_i)}$$

где:
> $\alpha = 1$ (параметр сглаживания)
> n_i — количество уникальных значений признака

Для непрерывных признаков предполагается нормальное распределение:

$$P(x_i|y_k) = \left(\frac{1}{2\pi\sigma^2}\right) \cdot \exp\left(-(x-\mu)^2/(2\sigma^2)\right)$$

где:

> μ — среднее признака для класса y_k

> σ — стандартное отклонение признака для класса y_k

Код программы

```
import numpy as np
import scipy.stats as sps
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn.datasets import load_iris, fetch_openml
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OrdinalEncoder

class NaiveBayes:
    def __init__(self, discrete_flags):
        """
        Конструктор
            принимает список флагов (True/False) для каждого признака,
            указывает, является ли признак дискретным
        """
        self.discrete_flags = discrete_flags
        self.class_probs = None
        self.feature_params = None
        self.classes = None

    def fit(self, X, y):
        """Обучение модели на тренировочных данных"""
        self.classes = np.unique(y)
        n_features = X.shape[1]

        self.class_probs = {c: np.mean(y == c) for c in self.classes}

        self.feature_params = {}

        for feature_idx in range(n_features):
            self.feature_params[feature_idx] = {}
```

```
            for c in self.classes:
                feature_values = X[y == c, feature_idx]

                if self.discrete_flags[feature_idx]:
                    unique, counts = np.unique(feature_values,
                        return_counts=True)
                    total = len(feature_values)
                    n_unique = len(unique)

                    # Сглаживание Лапласа
                    probs = {val: (cnt + 1)/(total + n_unique)
                            for val, cnt in zip(unique, counts)}
                    default_prob = 1/(total + n_unique)

                    self.feature_params[feature_idx][c] = {
                        'type': 'discrete',
                        'probs': probs,
                        'default_prob': default_prob
                    }
                else:
                    mean = np.mean(feature_values)
                    std = np.std(feature_values)
                    if std == 0: std = 1e-9

                    self.feature_params[feature_idx][c] = {
                        'type': 'continuous',
                        'mean': mean,
                        'std': std
                    }

    def predict(self, X):
        """Предсказание классов для новых данных"""
        predictions = []

        for obj in X:
            max_log_prob = -np.inf
            best_class = None

            for c in self.classes:
                log_prob = np.log(self.class_probs[c])
```

```
                for feature_idx in range(len(obj)):
                    params = self.feature_params[feature_idx][c]
                    val = obj[feature_idx]

                    if params['type'] == 'discrete':
                        if val in params['probs']:
                            log_prob += np.log(params['probs'][val])
                        else:
                            log_prob += np.log(params['default_prob'])
                    else:
                        log_prob += sps.norm.logpdf(val, loc=params['mean'], sca

                if log_prob > max_log_prob:
                    max_log_prob = log_prob
                    best_class = c

            predictions.append(best_class)

        return np.array(predictions)

def test_iris():
    """Тестирование на непрерывных данных (ирисы Фишера)"""
    iris = load_iris()
    X = iris.data
    y = iris.target

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=1/3, random_state=42)

    nb = NaiveBayes(discrete_flags=[False]*4)
    nb.fit(X_train, y_train)
    y_pred = nb.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    print(f"\nТестирование на Iris (непрерывные признаки):")
    print(f"Наивный Байес (реализация): {accuracy:.2f}")

    gnb = GaussianNB()
    gnb.fit(X_train, y_train)
    y_pred_sk = gnb.predict(X_test)
```

```
    accuracy_sk = accuracy_score(y_test, y_pred_sk)
    print(f"GaussianNB (sklearn): {accuracy_sk:.2f}")

def test_zoo():
    """Тестирование на дискретных данных (датасет Zoo)"""
    zoo = fetch_openml(name='zoo', version=1)
    X = zoo.data
    y = zoo.target

    encoder = OrdinalEncoder()
    X_encoded = encoder.fit_transform(X).astype(int)

    le = LabelEncoder()
    y_encoded = le.fit_transform(y)

    X_train, X_test, y_train, y_test = train_test_split(
        X_encoded, y_encoded, test_size=1/3, random_state=42)

    nb = NaiveBayes(discrete_flags=[True]*X_encoded.shape[1])
    nb.fit(X_train, y_train)
    y_pred = nb.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    print(f"\nТестирование на Zoo (дискретные признаки):")
    print(f"Наивный Байес (реализация): {accuracy:.2f}")

    mnb = MultinomialNB()
    mnb.fit(X_train, y_train)
    y_pred_sk = mnb.predict(X_test)
    accuracy_sk = accuracy_score(y_test, y_pred_sk)
    print(f"MultinomialNB (sklearn): {accuracy_sk:.2f}")

if __name__ == "__main__":
    test_iris()
    test_zoo()
```

Вывод программы

Для непрерывных данных на примере "ирисов Фишера"

```
Тестирование на Iris (непрерывные признаки):
Наивный Байес (реализация): 0.96
GaussianNB (sklearn): 0.96
```

Для дискретных данных на примере "зоопарка"

```
Тестирование на Zoo (дискретные признаки):
Наивный Байес (реализация): 0.88
MultinomialNB (sklearn): 0.88
```