Experiment 3 K-Nearest Neighbors for Classification

21230213093　陈现超

Problem description:

Given a training data set and a test data set, for k-nearest neighbors (k-NN) classification, we apply the Euclidean distance to measure the similarity between a pair of data. The k-NN classifier performs as the following steps.

For each test datum $\boldsymbol{x}$,

1. Among the training set, identify the k nearest neighbors of $\boldsymbol{x}$ (data most similar to $\boldsymbol{x}$).

2. Let c_i be the class most frequently found among these k nearest neighbors.

3. Label $\boldsymbol{x}$ with c_i.

Download Wine dataset from the UCI machine learning repository.

The sample size is 178. The first column is labeled, and the last thirteen list features.

```
# Wine 数据集
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
# 第一列为class，后十三个为特征
names = ['class', 'feature1','feature2','feature3','feature4','feature5','feature6',\
         'feature7','feature8','feature9','feature10','feature11','feature12','feature13',

data = pd.read_csv(url, names=names)
# data = pd.read_csv(url)
print(data)
```

```
     class  feature1  feature2  feature3  feature4  feature5  feature6  \
0        1     14.23      1.71      2.43      15.6       127      2.80
1        1     13.20      1.78      2.14      11.2       100      2.65
2        1     13.16      2.36      2.67      18.6       101      2.80
3        1     14.37      1.95      2.50      16.8       113      3.85
4        1     13.24      2.59      2.87      21.0       118      2.80
..     ...       ...       ...       ...       ...       ...       ...
173      3     13.71      5.65      2.45      20.5        95      1.68
174      3     13.40      3.91      2.48      23.0       102      1.80
175      3     13.27      4.28      2.26      20.0       120      1.59
176      3     13.17      2.59      2.37      20.0       120      1.65
177      3     14.13      4.10      2.74      24.5        96      2.05

     feature7  feature8  feature9  feature10  feature11  feature12  feature13
0        3.06      0.28      2.29       5.64       1.04       3.92       1065
1        2.76      0.26      1.28       4.38       1.05       3.40       1050
2        3.24      0.30      2.81       5.68       1.03       3.17       1185
3        3.49      0.24      2.18       7.80       0.86       3.45       1480
4        2.69      0.39      1.82       4.32       1.04       2.93        735
..        ...       ...       ...        ...        ...        ...        ...
173      0.61      0.52      1.06       7.70       0.64       1.74        740
174      0.75      0.43      1.41       7.30       0.70       1.56        750
175      0.69      0.43      1.35      10.20       0.59       1.56        835
176      0.68      0.53      1.46       9.30       0.60       1.62        840
177      0.76      0.56      1.35       9.20       0.61       1.60        560

[178 rows x 14 columns]
```

The Wine dataset has three labels that need to be scrambled before use.

```
# 将数据集转换为列表，以便打乱顺序
data_list = data.values.tolist()

# 打乱数据集的顺序
random.shuffle(data_list)

# 将打乱后的数据重新转换为DataFrame格式
shuffled_data = pd.DataFrame(data_list, columns=data.columns)

data = shuffled_data

print(data)
```

	class	feature1	feature2	feature3	feature4	feature5	feature6	\
0	1.0	14.23	1.71	2.43	15.6	127.0	2.80	
1	1.0	13.58	1.66	2.36	19.1	106.0	2.86	
2	3.0	12.53	5.51	2.64	25.0	96.0	1.79	
3	3.0	13.23	3.30	2.28	18.5	98.0	1.80	
4	3.0	12.36	3.83	2.38	21.0	88.0	2.30	
..	...	...	...	...	...	...	...	
173	2.0	12.51	1.73	1.98	20.5	85.0	2.20	
174	1.0	14.22	1.70	2.30	16.3	118.0	3.20	
175	3.0	12.25	4.72	2.54	21.0	89.0	1.38	
176	3.0	12.88	2.99	2.40	20.0	104.0	1.30	
177	2.0	12.16	1.61	2.31	22.8	90.0	1.78	

	feature7	feature8	feature9	feature10	feature11	feature12	feature13
0	3.06	0.28	2.29	5.64	1.04	3.92	1065.0
1	3.19	0.22	1.95	6.90	1.09	2.88	1515.0
2	0.60	0.63	1.10	5.00	0.82	1.69	515.0
3	0.83	0.61	1.87	10.52	0.56	1.51	675.0
4	0.92	0.50	1.04	7.65	0.56	1.58	520.0
..	...	...	...	...	...	...	...
173	1.92	0.32	1.48	2.94	1.04	3.57	672.0
174	3.00	0.26	2.03	6.38	0.94	3.31	970.0
175	0.47	0.53	0.80	3.85	0.75	1.27	720.0
176	1.22	0.24	0.83	5.40	0.74	1.42	530.0
177	1.69	0.43	1.56	2.45	1.33	2.26	495.0

[178 rows x 14 columns]

Euclidean distance is used here.

```
# 定义函数计算欧氏距离
def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2) ** 2))
```

Define a KNN classifier.

```
class KNNClassifier:
    def __init__(self, k=3):
        self.k = k

    def fit(self, X, y):
        self.X_train = X
        self.y_train = y

    def predict(self, X):
        y_pred = [self._predict(x) for x in X]
        return np.array(y_pred)

    def _predict(self, x):
        # 计算测试点与所有训练点的距离
        distances = [euclidean_distance(x, x_train) for x_train in self.X_train]
        # 获取最近的k个邻居的索引
        k_indices = np.argsort(distances)[:self.k]
        # 获取最近的k个邻居的标签
        k_nearest_labels = [self.y_train[i] for i in k_indices]
        # 返回出现频率最高的标签作为预测结果
        most_common = np.bincount(k_nearest_labels).argmax()
        return most_common
```

In this experiment, k=4 was selected, with 140 samples in the training set and 38 samples in the test set.

```
iris_data = data.iloc[:, 1:].values
iris_target = data.iloc[:, 0].values

# 划分训练集和测试集
X_train, X_test, y_train, y_test = iris_data[:140], iris_data[140:], \
                                   iris_target[:140], iris_target[140:]

# 初始化KNN分类器，选择k值
knn = KNNClassifier(k=4)

# 训练模型
knn.fit(X_train, y_train)

# 预测测试集
y_pred = knn.predict(X_test)
```

The classification accuracy was 0.74.

```
# 计算分类准确性
accuracy = np.mean(y_pred == y_test)
print(f"分类准确性：{accuracy}")
```

```
分类准确性：0.7368421052631579
```

Visualization of classification results.