hw5 knn

21302010042 侯斌洋

1. 数据分析

- span_pub.csv共有 4142 行，59 列，其中第一列为序号列，中间 57 列为特征列，最后一列为标签列。
- 特征列均为连续数值属性，标签列均为bool属性。
- 共有 4141 记录，其中 2521 个正样本，1620 个负样本。
- 数据中不存在缺失值，正负样本分布也较为均匀，适合用于决策树的训练。

2. 代码说明

2.1 KNN实现

```
class KNN:
  def __init__(self, k=5, distance='manhattan'):
    self.k = k
    self.distance = distance
    self.X_train = None
    self.Y_train = None

  def fit(self, X: list, Y: list) -> None:
    self.X_train = X
    self.Y_train = Y

  def predict(self, X: list) -> list:
    predicted_labels = [self._predict(x) for x in X]
    return np.array(predicted_labels)

  def _predict(self, x):
    # 计算距离
    if self.distance == 'euclidean':
      distances = [self._euclidean_distance(x, x_train) for x_train in
self.X_train]
    elif self.distance == 'manhattan':
      distances = [self._manhattan_distance(x, x_train) for x_train in
self.X_train]
    elif self.distance == 'chebyshev':
      distances = [self._chebyshev_distance(x, x_train) for x_train in
self.X_train]
    else:
      raise ValueError('Invalid distance')

    # 获取最近的k个样本
    k_indices = np.argsort(distances)[:self.k]

    # 获取k个样本的标签
    k_nearest_labels = [self.Y_train[i] for i in k_indices]
```

```
    # 返回出现次数最多的标签
    most_common = Counter(k_nearest_labels).most_common(1)
    return most_common[0][0]

  @staticmethod
  def _euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2) ** 2))

  @staticmethod
  def _manhattan_distance(x1, x2):
    return np.sum(np.abs(x1 - x2))

  @staticmethod
  def _chebyshev_distance(x1, x2):
    return np.max(np.abs(x1 - x2))
```

2.2 main.py

- 新增两个参数 `k` 和 `distance`，分别表示最近邻数和距离度量方式。参数的默认值为 `k=5` 和 `distance='manhattan'`。

```
def main(X: list, Y: list, test_x: list, k=5, distance='manhattan') -> list:
  knn = KNN(k, distance)
  knn.fit(np.array(X), np.array(Y))
  return knn.predict(np.array(test_x))
```

2.3 其他代码

```
# 读取数据
def load_csv(filename):...

# 划分训练测试集，默认训练集占70%，测试集占30%，每次划分时都随机打乱数据
def spit_data(data, test_rate):...

# 网格搜索,用于寻找最佳参数
def grid_search(data, test_rate):...

# 计算准确率
def compute_accuracy(predict_Y, test_Y):...

# 程序入口
if __name__ == '__main__':
  data = load_csv('./data/span_pub.csv')

  test_rate = 0.3

  # grid_search(data, test_rate)

  train_X, train_Y, test_X, test_Y = spit_data(data, test_rate)
  print(f'train_data_len: {len(train_X)}')
  print(f'test_data_len: {len(test_X)}')

  predict_Y = main(train_X, train_Y, test_X)
```

```
    accuracy = compute_accuracy(predict_Y, test_Y)

    print(f'accuracy: {accuracy}')
```

3. 运行结果

- 网格搜索的结果保存在`search_result.log`中，最佳参数为`k=5`和`distance='manhattan'`。以下为使用最佳参数训练的结果：

```
-----------------------------------------------
k: 5, distance: manhattan
accuracy0: 0.8407079646017699  time: 14.68901252746582
accuracy1: 0.8390989541432019  time: 13.670090198516846
accuracy2: 0.835076427996782  time: 13.092552900314331
average_accuracy: 0.8382944489139179
-----------------------------------------------
```

4. 优化

- KNN的实现按照标准的KNN算法。
- 采用了numpy的向量化计算，提高了计算效率。
- 使用网格搜索寻找最佳参数。搜索的参数为`k`和`distance`，分别表示最近邻数和距离度量方式。总计搜索了4种k值和3种距离度量方式，共12种组合。

```
search_space = {
        'k': [5, 10, 20, 30],
        'distance': ['manhattan', 'euclidean', 'chebyshev']
    }
```

- 网格搜索是机器学习中常用的调参方法之一。代码中也提供了网格搜索的实现，便于在不同的数据集上寻找最佳参数。
- 代码中`main.py`的默认参数为网格搜索得到的最佳参数。最终得到的准确率为`84%`左右。