

上海交通大学

SHANGHAI JIAO TONG UNIVERSITY



KNN 实验报告

学生姓名: 黄书航

学生学号: _____

专 业: _____

课程名称: _____

学院(系): _____

一、KNN 原理

KNN 算法($k - Nearest\ Neighbors$), 对一个输入, 从样本空间中选取 k 个最相似的样本, 将 k 个样本中大多数样本所属的那个类别作为这个输入的分类结果。

KNN 算法没有训练过程, 只有将训练用的正确分类样本和标签保存, 在分类时将输入与所有样本进行相似度比较和邻居类别统计, 在一些问题中可以取得比较好的效果。

二、实验

1. 补全 predict 函数

代码见图 1.

```
52 #####
53 # write your code here #
54 for i in range(len(test_data)):
55     lst = []
56     for j in range(len(self.train_data)):
57         # distance = np.sqrt(np.sum(np.square(test_data[i] - self.train_data[j])))
58         distance = self.get_distance(test_data[i], self.train_data[j])
59         lst.append((j, distance))
60     lst_odr = sorted(lst, key=lambda lst:lst[1], reverse=False)
61     res = [0] * max(20, self.k)
62     for num in range(0, self.k):
63         label = self.train_label[lst_odr[num][0]]
64         res[label] = res[label] + 1
65     predict_labels[i] = res.index(max(res))
66 #####
67 return np.array(predict_labels)
68
```

图 1

相似度的计算为输入向量和训练向量的 $manhattan$ 距离或 $euclidean$ 距离。

2. 实验

1) 第一个实验为原来的代码中自带的测试函数。

```
202 def test():
203     train_data, train_label, test_data, test_label = test_case()
204     algo = AlgorithmKNN(max_k=5, fold_num=5)
205     algo.fit(train_data, train_label)
206     print(np.mean(algo.predict(test_data) == test_label))
207
```

```

178 def test_case():
179     """
180     二维数据样例 D=2, N=100, M=2, 训练数据来自两个高斯分布的混合
181     :return: train_data(N,D), train_label(N,), test_data(M,D), test_label(M,)
182     """
183     mean = (1, 2)
184     cov = np.array([[73, 0], [0, 22]])
185     x = np.random.multivariate_normal(mean, cov, (80,))
186     mean = (16, -5)
187     cov = np.array([[21.2, 0], [0, 32.1]])
188     y = np.random.multivariate_normal(mean, cov, (20,))
189     train_data = np.concatenate([x, y])
190     train_label = np.concatenate([
191         np.zeros((80,), dtype=int),
192         np.ones((20,), dtype=int)
193     ])
194     test_data = np.array([
195         [3, 1], [3, 2], [1, -1], [2, 4], [7, 0], [6, 8], [-5, 0],
196         [18, -5], [5, -13], [16, -12], [17, -5], [13, -2], [19, -5], [17, -4]
197     ])
198     test_label = np.array([0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1])
199     return train_data, train_label, test_data, test_label
200

```

训练集为自动生成的高斯分布混合而成，训练集为自行增加的 11 个数据。

实验结果为：

```

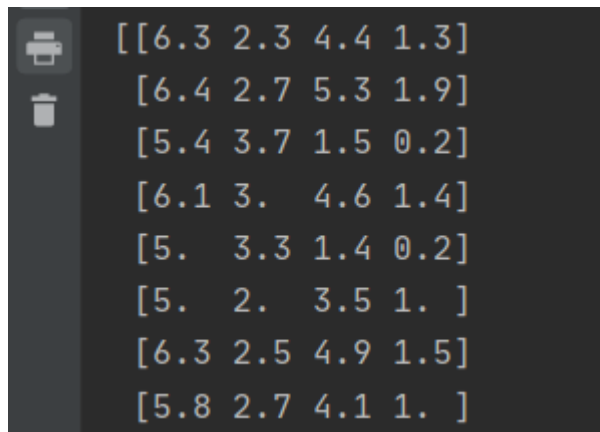
Run: source x
D:\Softwares\Anaconda3\envs\pr-assignment\python.exe D:/code/awesome-course
Running Test Function:
K=1, manhattan_distance, acc=0.87
K=1, euclidean_distance, acc=0.86
K=2, manhattan_distance, acc=0.88
K=2, euclidean_distance, acc=0.87
K=3, manhattan_distance, acc=0.87
K=3, euclidean_distance, acc=0.87
K=4, manhattan_distance, acc=0.87
K=4, euclidean_distance, acc=0.86
[choose K=2, distance_method=manhattan_distance]
0.9285714285714286

```

可以看到，KNN 的分类准确率比较高，k 从 1 到 4 变化后，在训练集准确率基本都在 0.87~0.88 之间，在测试集上表现也很好。

2) 第二个实验采用了知名的鸢尾花数据集。

鸢尾花数据集中每条数据包括四个数据，如下图所示：



```
[[6.3 2.3 4.4 1.3]
 [6.4 2.7 5.3 1.9]
 [5.4 3.7 1.5 0.2]
 [6.1 3. 4.6 1.4]
 [5. 3.3 1.4 0.2]
 [5. 2. 3.5 1. ]
 [6.3 2.5 4.9 1.5]
 [5.8 2.7 4.1 1. ]]
```

在本实验中设置训练集 120 条数据，测试集 30 条数据，K 从 1 到 14 变换，采用曼哈顿和欧几里得距离来衡量相似度，实验代码和实验结果如下：

```
209 def test_iris():
210     data = load_iris().data
211     label = load_iris().target
212     np.random.seed(10)
213     np.random.shuffle(data)
214     np.random.seed(10)
215     np.random.shuffle(label)
216
217     train_data = data[:120]
218     train_label = label[:120]
219     test_data = data[120:]
220     test_label = label[120:]
221     print(train_data)
222
223     algo = AlgorithmKNN(max_k=15, fold_num=8)
224     algo.fit(train_data, train_label)
225
226     print(np.mean(algo.predict(test_data) == test_label))
227
228
```

```
Run: source ×
K=1, manhattan_distance, acc=0.93
K=1, euclidean_distance, acc=0.95
K=2, manhattan_distance, acc=0.91
K=2, euclidean_distance, acc=0.94
K=3, manhattan_distance, acc=0.94
K=3, euclidean_distance, acc=0.95
K=4, manhattan_distance, acc=0.93
K=4, euclidean_distance, acc=0.94
K=5, manhattan_distance, acc=0.93
K=5, euclidean_distance, acc=0.95
K=6, manhattan_distance, acc=0.93
K=6, euclidean_distance, acc=0.94
K=7, manhattan_distance, acc=0.95
K=7, euclidean_distance, acc=0.96
K=8, manhattan_distance, acc=0.94
K=8, euclidean_distance, acc=0.95
K=9, manhattan_distance, acc=0.96
K=9, euclidean_distance, acc=0.97
K=10, manhattan_distance, acc=0.95
K=10, euclidean_distance, acc=0.94
K=11, manhattan_distance, acc=0.96
K=11, euclidean_distance, acc=0.95
K=12, manhattan_distance, acc=0.95
K=12, euclidean_distance, acc=0.94
K=13, manhattan_distance, acc=0.98
K=13, euclidean_distance, acc=0.97
K=14, manhattan_distance, acc=0.96
K=14, euclidean_distance, acc=0.97
[choose K=13, distance_method=manhattan_distance]
0.9333333333333333
```

可以看到用 KNN 来分类，准确度基本在于 0.91-0.98 之间准确度很高，采取 K=13 和曼哈顿距离，在测试集上准确度最高，在测试集上预测结果也比较好。