实验报告

1 基本思路

代码指路链接:

林胜峰的Git Hub

先看 `main()` 函数

```
def main(k):
    global accuracy
    # step 1. train()
    train()
    # step 2. validate(i)
    for i in range(1, k + 1):
        accuracy.append(validate(i))
    # step 3. draw()
    draw(k, accuracy)
    # step 4. 获取验证效果最好的k, 即 accuracy最大值的下标 + 1
    index_max = accuracy.index(max(accuracy)) + 1
    # step 5. 预测, 并计算预测精度
    do_pre(index_max)


if __name__ == "__main__":
    main(5)  # 超参数从1到5试验一遍

```

1.1 读取训练数据集并划分训练集和验证集 `train()`

```
# 这个函数的作用有:
# 1. 读取 train.csv 中的数据
# 2. 划分数据集(3:1 = 训练数据集:验证数据集)
def train():
    global training_labels, training_vectors, validate_labels,
validate_vectors
    with open('train.csv') as training_csv:
        training_reader = csv.reader(training_csv, delimiter=' ',
quotechar='|')
        # get rid of header of csv
        next(training_reader, None)
```

```
        for row in training_reader:
            row = [int(i) for i in row[0].split(',')]
            training_labels.append(row[0])  # 把标签加入训练标签列标中
            training_vectors.append(row[1:])  # 把数据加入训练数据集中
        # 划分数据集, 训练:验证 = 3:1, 完全随机划分
        training_vectors, validate_vectors, training_labels,
validate_labels = \
            train_test_split(training_vectors, training_labels,
test_size=0.25, random_state=0)
    print("Done training!")
```

1.2 超参数设置为k,并用验证集验证 `validate(k)`

```
# 验证, k表示knn的超参数k
def validate(k):
    global validate_vectors, validate_labels, validate_temp_labels
    i = 0
    validate_temp_labels = [-1] * len(validate_vectors)
    for row in validate_vectors:
        min_dist = [sys.maxsize] * k  # 长度为k的最小距离列表，最小距离计算方法
在dist()方法中
        min_dist_label = [-1] * k  # 长度为k的标签列表，存放可能的标签
        for count, (vec, num) in enumerate(zip(training_vectors,
training_labels)):  # 遍历训练集
            if count % 2000 == 0:  # 其实只有 count 为 0 的时候才生效
                print("Validating on No." + str(i))
            d = dist(row, vec)  # 计算 row(即当前数据) 与 vec(当前测试数据) 的
距离
            if d < max(min_dist):  # 如果距离比 min_dist 中的最大值小
                min_dist_label[min_dist.index(max(min_dist))] = num  # 替换
对应最大距离的标签为vec的标签num
                min_dist[min_dist.index(max(min_dist))] = d  # 替换最大距离为
当前距离d
        validate_temp_labels[i] = get_most_label(min_dist, min_dist_label)
 # 检查最有可能的标签
        i += 1
    print("Validate done by KNN, where k = " + str(k))
    # 返回当前k获取的精度
    return metrics.accuracy_score(validate_labels, validate_temp_labels)
```

1.2.1 计算两个向量距离的函数 `dist(v1, v2)`

```python
# 计算两个向量距离的函数，其实，可以理解为 sum = abs(v1[i] - v2[i]), for i in range(len(v1))，注意 v1与v2长度必相同
def dist(vector_one, vector_two):
    d = 0
    for i, j in zip(vector_one, vector_two):
        d += abs(i - j)
    return d
```

1.3 绘制评价图 `draw()`

```python
# 绘制评价图，可以是精度也可以是f1，两个参数中必定有一个为True，另一个为 False
def draw(k, score, accuracy_score=True, f1_score=False):
    x = [i for i in range(1, k + 1)]
    name = ''
    if accuracy_score and not f1_score:
        name = 'Accuracy'
    if f1_score and not accuracy_score:
        name = 'F1'
    plt.plot(x, score, label=name + ' Score', linewidth=3, color='r', marker='o')
    plt.xlabel('number of hyper parameter k')
    plt.ylabel(name + ' Score')
    plt.title(name + ' Score of k from 1 to 5')
    plt.show()
```

1.4 测试 `do_pre(k)`

```python
# 设置超参数为k，然后进行predict(k)，再根据真实的结果(保存在truth.csv)计算超参数为k得到的精度
def do_pre(k):
    global truth_labels, learning_labels
    predict(k)
    truth_labels = get_labels('truth.csv', truth_labels)
    result = metrics.accuracy_score(truth_labels, learning_labels)
    print(result)
```

1.4.1 测试的核心函数 `predict(k)`

```python
# 预测，逻辑大致与validate()相同
def predict(k):
    global training_labels, training_vectors, learning_labels, learning_vectors
    with open('test.csv') as test_csv:  # 测试数据在 test.csv
```

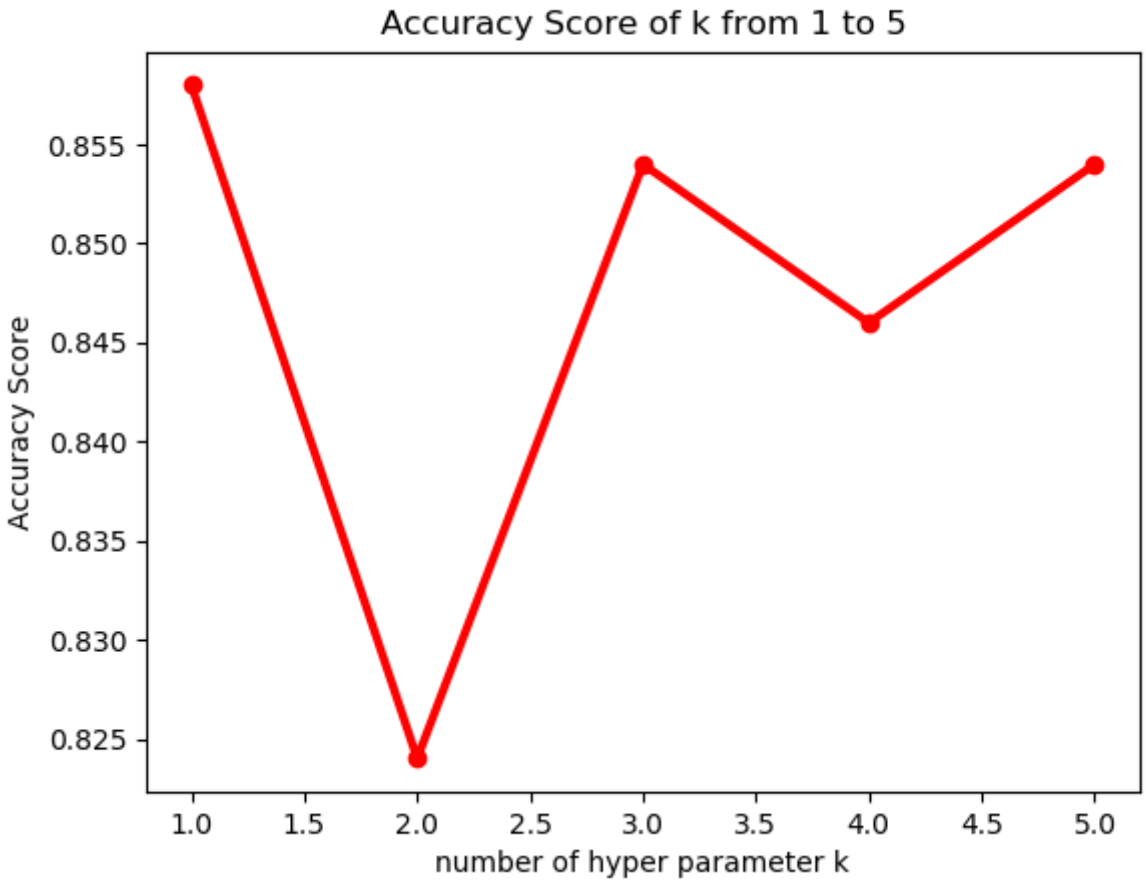
```
        learning_reader = csv.reader(test_csv, delimiter=' ',
quotechar='|')
        next(learning_reader, None)
        learning_vectors = list(learning_reader)
        i = 0
        learning_labels = [-1] * len(learning_vectors)
        for row in learning_vectors:
            row = [int(j) for j in row[0].split(',')]
            min_dist = [sys.maxsize] * k
            min_dist_label = [-1] * k
            closest_num = -1
            for count, (vec, num) in enumerate(zip(training_vectors,
training_labels)):
                if count % 10000 == 0:
                    print("Working on NO." + str(i))
                d = dist(row, vec)
                if d < max(min_dist):
                    min_dist_label[min_dist.index(max(min_dist))] = num
                    min_dist[min_dist.index(max(min_dist))] = d
                    closest_num = get_most_label(min_dist, min_dist_label)
            learning_labels[i] = closest_num
            i += 1
    print("KNN Done!")
```

1.4.2 获取真实标签的函数 `get_labels(f, labels)`

```
# 从csv文件获取标签的函数
def get_labels(csv_file, labels):
    with open(csv_file) as temp_csv:
        temp_reader = csv.reader(temp_csv, delimiter=' ', quotechar='|')
        next(temp_reader, None)
        for row in temp_reader:
            row = [int(i) for i in row[0].split(',')]
            labels.append(row[0])
    print("Get " + csv_file + " labels done!")
    return labels
```

2 实验结果

针对这个数据集，我将 `train` 数据随机 `3:1` 划分为 `train` 和 `validate`，并将 `k` 从 `1到5` 测试了一遍，每个k得到的精度(accuracy)图如下:



由图可知，`k=1` 时，精度值最高，所以设置超参数 `k=1` 进行测试。

测试结果如下：

```
k: 1
accuracy: 0.8866171003717472
```

3 全部代码

```python
import csv
import sys
import math
from sklearn.model_selection import train_test_split
from sklearn import metrics
import random
import matplotlib.pyplot as plt

# 训练标签
training_labels = []
# 训练数据
training_vectors = []

# 学习(预测)标签
learning_labels = []
```

```python
# 学习数据
learning_vectors = []

# 验证标签
validate_labels = []
# 验证数据
validate_vectors = []
# 验证预测标签: 即 由knn算法为验证集预测的标签，它与真实的验证集标签可以用来判断超参数k的好坏
validate_temp_labels = []
# 精度: 存放每个k进行实验后的精度
accuracy = []

# 学习数据的真实标签
truth_labels = []


# 这个函数用于获取一个数据可能的标签 这个列表中占比最多的标签，如果有一样多的标签，则随机取
def get_most_label(dists, labels):
    # 0-9所有数的出现次数，以字典的形式保存
    times = {0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0}
    for order in range(len(dists)):
        if labels[order] != -1:  # -1是无效标签，要排除
            times[labels[order]] += 1
    tem = sorted(times.items(), key=lambda x: x[1], reverse=True)  # 倒序排序，也就是从大到小排序
    most = [tem[0][0]]
    for i in range(1, len(tem)):
        if tem[i][1] == tem[0][1]:
            most.append(tem[i][0])
        else:
            break
    if len(most) == 1:
        return most[0]
    else:
        return most[random.randint(0, len(most) - 1)]


# 这个函数的作用有:
# 1. 读取 train.csv 中的数据
# 2. 划分数据集(3:1 = 训练数据集:验证数据集)
def train():
    global training_labels, training_vectors, validate_labels, validate_vectors
    with open('train.csv') as training_csv:
        training_reader = csv.reader(training_csv, delimiter=' ', quotechar='|')
        # get rid of header of csv
        next(training_reader, None)
        for row in training_reader:
            row = [int(i) for i in row[0].split(',')]
```

```
63              training_labels.append(row[0])  # 把标签加入训练标签列标中
64              training_vectors.append(row[1:])  # 把数据加入训练数据集中
65          # 划分数据集，训练:验证 = 3:1，完全随机划分
66          training_vectors, validate_vectors, training_labels,
    validate_labels = \
67              train_test_split(training_vectors, training_labels,
    test_size=0.25, random_state=0)
68      print("Done training!")
69
70
71  # 验证，k表示knn的超参数k
72  def validate(k):
73      global validate_vectors, validate_labels, validate_temp_labels
74      i = 0
75      validate_temp_labels = [-1] * len(validate_vectors)
76      for row in validate_vectors:
77          min_dist = [sys.maxsize] * k  # 长度为k的最小距离列表，最小距离计算方法
    在dist()方法中
78          min_dist_label = [-1] * k  # 长度为k的标签列表，存放可能的标签
79          for count, (vec, num) in enumerate(zip(training_vectors,
    training_labels)):  # 遍历训练集
80              if count % 2000 == 0:  # 其实只有 count 为 0 的时候才生效
81                  print("Validating on No." + str(i))
82              d = dist(row, vec)  # 计算 row(即当前数据) 与 vec(当前测试数据) 的
    距离
83              if d < max(min_dist):  # 如果距离比 min_dist 中的最大值小
84                  min_dist_label[min_dist.index(max(min_dist))] = num  # 替换
    对应最大距离的标签为vec的标签num
85                  min_dist[min_dist.index(max(min_dist))] = d  # 替换最大距离
    为当前距离d
86          validate_temp_labels[i] = get_most_label(min_dist, min_dist_label)
     # 检查最有可能的标签
87          i += 1
88      print("Validate done by KNN, where k = " + str(k))
89      # 返回当前k获取的精度
90      return metrics.accuracy_score(validate_labels, validate_temp_labels)
91
92  # 预测，逻辑大致与validate()相同
93  def predict(k):
94      global training_labels, training_vectors, learning_labels,
    learning_vectors
95      with open('test.csv') as test_csv:  # 测试数据在 test.csv
96          learning_reader = csv.reader(test_csv, delimiter=' ',
    quotechar='|')
97          next(learning_reader, None)
98          learning_vectors = list(learning_reader)
99          i = 0
100         learning_labels = [-1] * len(learning_vectors)
101         for row in learning_vectors:
102             row = [int(j) for j in row[0].split(',')]
103             min_dist = [sys.maxsize] * k
104             min_dist_label = [-1] * k
```

```
            closest_num = -1
            for count, (vec, num) in enumerate(zip(training_vectors, training_labels)):
                if count % 10000 == 0:
                    print("Working on NO." + str(i))
                d = dist(row, vec)
                if d < max(min_dist):
                    min_dist_label[min_dist.index(max(min_dist))] = num
                    min_dist[min_dist.index(max(min_dist))] = d
                    closest_num = get_most_label(min_dist, min_dist_label)
            learning_labels[i] = closest_num
            i += 1
    print("KNN Done!")


# 把超参数k获取的结果写入一个csv中
def write(k):
    global learning_labels
    file_k = 'answers' + str(k) + '.csv'
    with open(file_k, 'w', newline='') as csvfile:
        answer_writer = csv.writer(csvfile, delimiter=' ', quotechar='|', quoting=csv.QUOTE_MINIMAL)
        answer_writer.writerow(['Answers'])
        for label in learning_labels:
            answer_writer.writerow(str(label))
    print("Done writing CSV!")


# 计算两个向量距离的函数，其实，可以理解为 sum = abs(v1[i] - v2[i]), for i in range(len(v1)), 注意 v1与v2长度必相同
def dist(vector_one, vector_two):
    d = 0
    for i, j in zip(vector_one, vector_two):
        d += abs(i - j)
    return d


# 从csv文件获取标签的函数
def get_labels(csv_file, labels):
    with open(csv_file) as temp_csv:
        temp_reader = csv.reader(temp_csv, delimiter=' ', quotechar='|')
        next(temp_reader, None)
        for row in temp_reader:
            row = [int(i) for i in row[0].split(',')]
            labels.append(row[0])
    print("Get " + csv_file + " labels done!")
    return labels


# 绘制评价图，可以是精度也可以是f1，两个参数中必定有一个为True，另一个为 False
def draw(k, score, accuracy_score=True, f1_score=False):
    x = [i for i in range(1, k + 1)]
```

```
154     name = ''
155     if accuracy_score and not f1_score:
156         name = 'Accuracy'
157     if f1_score and not accuracy_score:
158         name = 'F1'
159     plt.plot(x, score, label=name + ' Score', linewidth=3, color='r',
    marker='o')
160     plt.xlabel('number of hyper parameter k')
161     plt.ylabel(name + ' Score')
162     plt.title(name + ' Score of k from 1 to 5')
163     plt.show()
164
165
166 # 设置超参数为k, 然后进行predict(k), 再根据真实的结果(保存在truth.csv)计算超参数
    为k得到的精度
167 def do_pre(k):
168     global truth_labels, learning_labels
169     predict(k)
170     truth_labels = get_labels('truth.csv', truth_labels)
171     result = metrics.accuracy_score(truth_labels, learning_labels)
172     print(result)
173
174
175 def main(k):
176     global accuracy
177     # step 1. train()
178     train()
179     # step 2. validate(i)
180     for i in range(1, k + 1):
181         accuracy.append(validate(i))
182     # step 3. draw()
183     draw(k, accuracy)
184     # step 4. 获取验证效果最好的k, 即 accuracy最大值的下标 + 1
185     index_max = accuracy.index(max(accuracy)) + 1
186     # step 5. 预测, 并计算预测精度
187     do_pre(index_max)
188
189
190 if __name__ == "__main__":
191     main(5)
```