AI第一次实验—KNN的实现

18308133 刘显彬 计科二班

一、实验原理

1.KNN原理

KNN是一种邻近预测算法，也属于监督学习的一种（需要有标记的数据），通过收集训练数据作为基准数据，当需要评估某个（些）测试数据时，就逐一计算其与基准数据的相似度，取出最为相似的一组（k个）基准数据，通过某些权重加成（如测试数据和这组基准数据的相似度作为权重）以及依照某种规则（比如投票：选出现标记最多的一个）选出其中一个标记作为该测试数据的预测结果。其中：

带有标记的**数据**可以用其特征值+标记值组成，而测试数据只有特征值，特征可有多个。

相似度可以用数据间的（各种）距离来衡量：如果数据的特征值可以量化的话，我们就可以利用量化后的排列有序的特征值（特征向量）来表征该数据的属性， 此时我们就可以对两个特征向量求出各种距离：曼哈顿距离、欧式距离、余弦距离等：其中各种距离公式如下：

$$假设：x_i, x_j是两个量化后的特征向量，特征个数（维度）为n，则有：$$

$$L_p(x_i, x_j) = \{\sum_{k=1}^{n} |x_{ik} - x_{jk}|^p\}^{\frac{1}{p}}，其中p = 1为曼哈顿距离，2为欧式距离$$

$$余弦距离公式：cos(x_i, x_j) = \frac{x_i * x_j}{|x_i|\,|x_j|}$$

2.利用KNN模型的流程

2.1 由上面的分析，我们可以总结出运用KNN模型需要的实现：

1. KNN需要**存储**（大量的）已标记的数据用于训练模型
2. 数据处理：需要对数据的特征进行**量化**便于分析数据间的距离度量，特别地，在这个实验中选用的是**tf-idf**编码的方式进行数据处理
3. 距离函数：选择合适的距离函数可以加强特征的利用（个人认为：比如欧式距离弱化了各个特征的区别，在空间上是一个球面，即各个方向的类似），实验中同时完成了上述提到的三种距离
4. 预测函数：将测试数据传入KNN模型，利用距离函数计算出测试数据和所有训练数据的距离，取出与测试数据最相似的k个数据，根据k个标记中最多的一个作为测试数据的标记。

2.2 其中，KNN模型需要调整两个超参数：k和距离函数类型，后面会分析到不同参数值对模型的影响，在正式将我们的模型用于预测之前，我们需要根据某些流程选出可能的、最合适的模型参数：

1. 将已经标记好的数据集划出一部分用于考察参数选择的效果，这部分数据称为验证集(Validation Set)，其余部分（**不相交**）就是我们的训练数据集(train set)；**对所有数据进行预处理（量化）**
2. 设定好KNN模型的超参数（k和距离函数），将训练集传入模型存储；将验证集数据传入模型并调用预测函数，预测这些数据的**标记**，并与真实标记进行对比，计算出预测的正确率，正确率即可作为这个超参数配置下的模型的评估分数；
3. 重复（2），找出表现最好的一组超参数配置，这组配置即可以用于测试数据的预测

二、伪代码

$1.TF-IDF:$

```
def TF-IDF(sentences):
  words_doc = get_different_word(sentences)     # 数出句子中所有不同的单词
  n = len(words_doc)  # 词表长度
  k = len(sentences)
  sentence_one_hot = int[k][n]
  for sentence_i in sentences:
    sentence_one_hot[i] = [0]*n
    # 转换成one-hot编码，其中index求取word_j在词表的位置
    for word_j in sentence:
      sentence_one_hot[i][index(words_doc, word_j)] += 1
  TF = sentence_one_hot / sum(sentence_one_hot)   # 归一化
  # 完成TF构建

  # IDF:
  words_count = [0]*n
  for sentence in sentences:
    for word in sentence:
      if word not exist since this cycle:     # 如果该单词在此句子没出现过，则计数+1
        words_count[index(word)] += 1
  IDF = log(n / (1+ words_count))             # 转化成idf
  return TF*IDF
```

$2.Predict\_with\_k:$

```
def predict(X, k, distance):           # 需要：测试数据，距离函数，k值
  dist = distance(training_data, X)    # 根据距离公式计算测试数据x和训练数据的距离
  K_index = min_K(dist, k)             # 获取前K个最近的训练数据
  K_labels= training_data[K_index]     # 取得他们的label
  return countMax(K_labels)            # 找出最多的一个并返回
```

$3.improve\_predict\_for\_regression:$

```
def predict_regression(X, K, distance):
  dist = distance(training_data, X)    # 根据距离公式计算测试数据X和训练数据的距离
  K_index = min_K(dist, k)             # 获取前K个最近的训练数据
  K_labels= training_data[K_index]     # 取得他们的label
  K_dist = dist[K_index]               # 取得他们的距离
  factor = sum(1/K_dist)               # 放缩因子为：距离倒数和
  return sum(K_label, weight=1/ K_dist)*factor  # 乘以权重相加
```

4.整体流程

```
model=KNN()
test_data, train_data, val_data = TF-IDF(sentences) # 转化为TF-IDF矩阵
model.train(train_data)
for param in params_possible:                       # 找最佳参数
  predict_result=predict(val_data, param)           # 预测val data 结果
  performance[param] = evalution(predict_result)    # 评估这组参数

findBest(performance) # 找到最佳参数
test_result = model.predict(test_data, bestParam)   # 预测test data的结果并存储
save(test_result)
```

三、关键代码展示

1.TF-IDF

1.1将句子根据词汇表编码成TF向量，其中词汇表存储了每一个单词对应的one-hot编码非零的下标，tf矩阵加上一个偏移是因为后面两个实验进行归一化的时候会出现分母为0的情况，所以加上一个小的偏移保证非零，本个实验中偏移值是0。

```
def encoder_tf(content, vocab, epsilon=1e-4):
    tf = np.zeros((len(content), len(vocab)))
    for i in range(len(content)):
        for word in content[i]:
            if word in vocab:
                tf[i][vocab[word]] += 1
    tf += epsilon
    tf /= np.sum(tf, axis=1, keepdims=1)
    return tf
```

1.2 IDF：统计词汇出现的文章数

```
for word in line:
    if word not in vocab:
        vocab[word] = len(vocab)
    if word not in exist:   # word在这个句子还没出现过
        index = vocab[word]       # 取出word对应下标
        idf[index] = idf[index] + 1 if index in idf else 1  # 计数+1
        exist.add(word)
content.append(line)
```

2.distances

2.1 p=1，街区距离（曼哈顿距离）直接对应位置相减绝对值求和；

```
if distType == 1:
    res = []
    for i in range(len(X)):
        res.append(np.sum(np.abs(self.X-X[i]), axis=1))
    return np.array(res)
```

2.2 p=2，欧式距离*向量版，向量版原理在稍后给出

```
elif distType == 2:
    res = np.sum(X**2, axis=1, keepdims=1) + np.sum(self.X**2, axis=1) - 2.0*X.dot(self.X.T)
    res = np.maximum(res, 0)
    return np.sqrt(res)
```

(parameter) keepdims: bool

```
                # because the float computing could make the res smal
                # or the error: RuntimeWarning: invalid value encount
```

2.3 余弦距离，非矩阵运算形式

```
# mod_train_X = np.linalg.norm(self.X, axis=1)
# for i in range(len(X)):
#     norm_Xi = np.linalg.norm(X[i])
#     res.append(X[i].dot(self.X.T)/(norm_Xi*mod_train_X))
# return 1-np.array(res)
```

完全向量化：

```
# in vector way
self_X_norm = self.X / np.linalg.norm(self.X, axis=1, keepdims=1)
X_norm = X / np.linalg.norm(X, axis=1, keepdims=1)
return 1 - X_norm.dot(self_X_norm.T)
```

3.classification

3.1 预测：得到距离最小的k个样本，并利用bincount来对不同的label计数，利用argmax取出数目最多的label

```
dist = self.distance(X, lossType)
dist_K = np.argsort(dist)[:,:self.K]
# Y = np.argmax(dist_K, axis=1)
Y_K = self.y[dist_K]
for i in range(len(Y_K)):
    # get the label of largest number in the K labels
    # use bincount for counting the max
    maxp = np.bincount(Y_K[i])
    Y[i] = argmax(maxp)
Y = Y.astype(int).reshape(-1)
return self.y[Y]
```

3.2 训练：调整不同的超参数，利用validation set对不同参数配置进行准确率评估，选择出最优的一组参数对test Set进行预测

```
for k in ks:
    for losstype in L:
        model.train(train_set, label_tr)        # save the training data
        y_pred = model.predict(val_set,k=k, lossType=losstype)  # predict
        thisac = np.mean(y_pred == label_v)     # calculate the accuracy
        print("the accuracy of config(k:%d, losstype:%d) is %f" %(k, losstype, thisac))
        if thisac > best_ac:
            best_ac = thisac                    # update the best performance
            best_config = (k, losstype)
        ac[(k, losstype)] = thisac              # record the evaluation for the config
```

4.regression

4.1 修改predict：取到最近的K个样本的**距离**，令他们的倒数作为权重，乘以他们的label值再相加，得到测试数据的估测向量（因此在这里的label是一个N维向量，N代表了label的种类，值代表了label的概率），并除以K个距离的倒数和，将该估测向量归一化成概率。

（思考题中展示了为什么如此操作可以得到一个和为1的估测向量）。

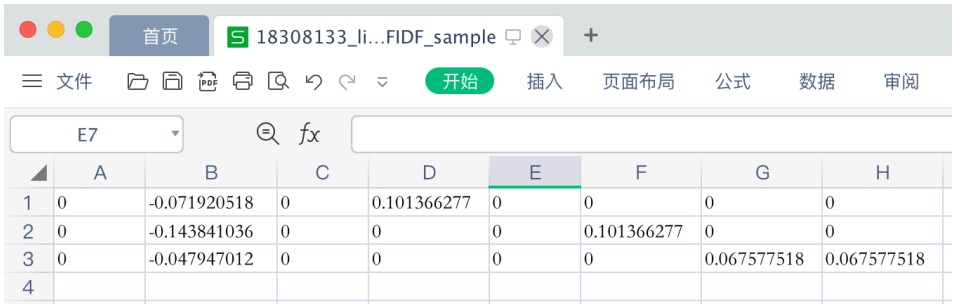
```
        Y_K = self.y[dist_K]        # shape : <N, K, C>
        for i in range(len(Y_K)):
            dist_k_i = dist[i][dist_K[i]].reshape(self.K,1)    # get the K distances
            # note that np.sum should point out the sum for the columns,
            # and the Denominator : " 1/dist_k_i" used to make the sum of prob to be 1
            Y.append(np.sum(Y_K[i] / dist_k_i, axis=0) / np.sum(1/dist_k_i)) # of shape <K, C>
                # Y_K[i][k] / distance_k

        return np.array(Y)
```

四、结果展示

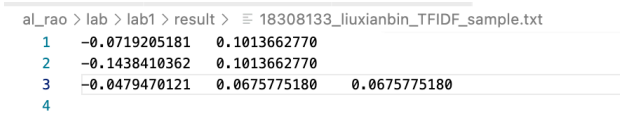
1.TF-IDF

1.1 result of sample: (path:/result/18308133_liuxianbin_TFIDF_sample.csv)



	A	B	C	D	E	F	G	H
1	0	-0.071920518	0	0.101366277	0	0	0	0
2	0	-0.143841036	0	0	0	0.101366277	0	0
3	0	-0.047947012	0	0	0	0	0.067577518	0.067577518
4								

删除多余的0，得到：



与助教师兄/师姐给的参考一致：

19881234_Sample_TFIDF.txt.txt

```
├0.0719205 0.101366
-0.143841 0.101366
-0.047947 0.0675775 0.0675775
```

1.2 result of test set:(path:/result/18308133_liuxianbin_TFIDF.csv)

semeval.txt转化成tf-idf结果如下：

knn.py M | regression.py M | tfidf.py M | 18308133_liuxianbin_TFIDF.txt U

al_rao > lab > lab1 > result > 18308133_liuxianbin_TFIDF.txt

1	1.0724244198	1.0048469018	0.8217448537	0.5393122335	1.0048469018	0.7604573903
2	1.4353498346	1.6086366297	0.5829757884	1.6086366297		
3	0.7604573903	0.8893223717	1.0724244198	0.8413753596	0.5535052016	0.9197092978
4	2.1448488396	1.1907818793	1.9137997794			
5	0.9568998897	0.6727752077	1.0724244198	0.9197092978	0.8041847677	1.0724244198
6	0.8893223717	1.0724244198	0.8893223717	0.4683675976	0.7604573903	1.0724244198
7	0.8302578024	1.6086366297	1.5072703527	1.6086366297		
8	0.9197092978	1.0724244198	0.7366072497	0.4000509801	0.8636305917	0.9568998897
9	1.9137997794	2.1448488396	1.6434897073			
10	0.6379332598	0.7149496132	0.6698979345	0.4591068251	0.2162122388	0.6698979345
11	0.3595414890	0.6379332598	0.5928815811	0.2323045663	0.3740548426	0.7149496132
12	0.1945910149	0.2090741097	0.5181783550	0.6434546519	0.6029081411	0.5741399338
13	0.7604573903	0.2936196141	0.8041847677	0.6972091200	1.0724244198	0.7157467259
14	0.7604573903	1.0048469018	1.4962121232	1.0048469018	1.0724244198	
15	0.5335934230	0.6434546519	0.6029081411	0.3321031210	0.6434546519	0.3693706495
16	0.9125488684	0.8263922852	0.3891820298	1.1482798676	1.1482798676	
17	0.1793771657	0.3509803340	0.4104564792	0.3216524765	0.2589610449	0.3985987346
18	0.5703430427	0.2613426371	0.2202147105	0.5368100444	0.6477229438	0.5610795462
19	0.4683675976	0.8893223717	1.0048469018	1.0724244198	1.0724244198	1.0724244198
20	0.2613426371	0.6669917788	0.7536351763	0.7536351763	0.5803483812	0.5296652427

2.classification

2.1 result of test set:

调整超参数时的输出，输出数据是对应不同k和距离函数下的准确率，其中losstype为1，2，3分别表示曼哈顿距离、欧式距离、余弦距离，可以看出对该实验validation Set而言，k=7，距离函数取余弦距离时，KNN模型表现最好，准确率可以达到0.463（稍后将分析为什么不同k和距离函数会导致模型的不一样）

```
liuxb@liuxianbindeMacBook-Air 2021_au % /usr/local/bin/python3 /Users/liuxb/2021_au/al_rao/lab/lab1
/code/classification.py
the accuracy of config(k:6, losstype:1) is 0.395498
the accuracy of config(k:6, losstype:2) is 0.372990
the accuracy of config(k:6, losstype:3) is 0.443730
the accuracy of config(k:7, losstype:1) is 0.421222
the accuracy of config(k:7, losstype:2) is 0.395498
the accuracy of config(k:7, losstype:3) is 0.463023
the accuracy of config(k:8, losstype:1) is 0.411576
the accuracy of config(k:8, losstype:2) is 0.389068
the accuracy of config(k:8, losstype:3) is 0.450161
the accuracy of config(k:9, losstype:1) is 0.414791
the accuracy of config(k:9, losstype:2) is 0.366559
the accuracy of config(k:9, losstype:3) is 0.434084
the accuracy of config(k:10, losstype:1) is 0.398714
the accuracy of config(k:10, losstype:2) is 0.389068
the accuracy of config(k:10, losstype:3) is 0.443730
the accuracy of config(k:12, losstype:1) is 0.427653
the accuracy of config(k:12, losstype:2) is 0.363344
the accuracy of config(k:12, losstype:3) is 0.421222

 the best config is k(7), losstype(3) with accuracy 0.463023
```

下面是test_set的预测结果：(path: /result/18308133_liuxianbin_classification.csv)

classification.py M | 18308133_liuxianbin_classification.csv M | knn.p

al_rao > lab > lab1 > result > 18308133_liuxianbin_classification.csv

```
textid,Words (split by space),label
1,who is prince frederic von anhalt,joy
2,prestige has magic touch,joy
3,study female seals picky about mates,joy
4,no e book for harry potter vii,joy
5,blair apologises over friendly fire inquest,sad
6,vegetables may boost brain power in older adults,joy
7,afghan forces retake town that was overrun by taliban,sad
8,skip the showers male sweat turns women on study says,joy
9,made in china irks some burberry shoppers,joy
10,britain to restrict immigrants from new eu members,fear
11,canadian breakthrough offers hope on autism,joy
12,russia to strengthen its military muscle,fear
13,alzheimer s drugs offer no help study finds,joy
14,uk police slammed over terror raid,fear
15,no rejects police state claim,fear
16,oprah announces new book club pick,joy
17,smith can t be buried until hearing,sad
18,african nation hopes whoopi can help,joy
19,tourism lags in bush s hometown,joy
20,air france klm profit rises,sad
21,au regrets sudan s expulsion of un envoy,joy
22,taiwan s mr clean indicted steps down,joy
23,martian life could have evaded detection by viking landers,joy
24,turner pays for boston bombing,joy
25,serena misses bangalore tournament,joy
26,attorneys point fingers in fight between ryan o neal and son,joy
```

3.regression

3.1 result of validation set:

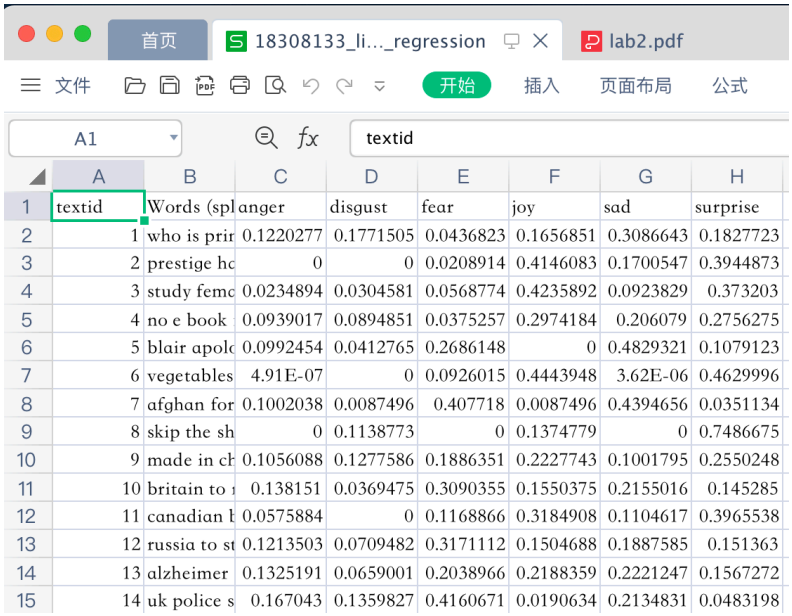
这里由于label不再是单一的离散值，而是各种label的估计概率，因此这里改用相关系数来代替准确率作为模型评价，这里的最佳相关系数为0.406（对六个label取平均），这里导出了最佳的validation Set的预测结果（**path: /result/validation_predict.csv)**，放入评估相关系数的EXCEL表格的结果见第二个图

```
liuxb@liuxianbindeMacBook-Air 2021_au % /usr/local/bin/python3 /Users/liuxb/2021_au/al_rao/lab/lab1
/code/regression.py
the cor of config(k:5, losstype:1) is 0.341042
the cor of config(k:5, losstype:2) is 0.272781
the cor of config(k:5, losstype:3) is 0.406785
the cor of config(k:6, losstype:1) is 0.334269
the cor of config(k:6, losstype:2) is 0.265983
the cor of config(k:6, losstype:3) is 0.401650
the cor of config(k:7, losstype:1) is 0.336422
the cor of config(k:7, losstype:2) is 0.264831
the cor of config(k:7, losstype:3) is 0.400621
the cor of config(k:8, losstype:1) is 0.337660
the cor of config(k:8, losstype:2) is 0.267388
the cor of config(k:8, losstype:3) is 0.405399
the cor of config(k:9, losstype:1) is 0.335885
the cor of config(k:9, losstype:2) is 0.272535
the cor of config(k:9, losstype:3) is 0.405024
the cor of config(k:10, losstype:1) is 0.338648
the cor of config(k:10, losstype:2) is 0.278150
the cor of config(k:10, losstype:3) is 0.406749
best config for validation set is: k:5, losstype:3, best cor:0.406785
already save the test result into file:/Users/liuxb/2021_au/al_rao/lab/lab1/result/18308133_liuxian
bin_regression.csv
```

anger	disgust	fear	joy	sad	surprise			anger
0.0354	0.06471	0.25502	0.34494	0.08682	0.2131		r	0.407067565
0.09065	0.02367	0.20283	0.20726	0.21647	0.25915		average	0.406785384
0.07816	0.04095	0.12645	0.27173	0.05842	0.42427		evaluation	低度相关 666
0.08689	0.0815	0.22128	0.10917	0.33596	0.16524			
0.03974	0.02646	0.17027	0.33794	0.20086	0.22474			
0.25832	0.10128	0.11484	0.06474	0.22295	0.23779			
0.1639	0.09025	0.29881	0.07744	0.24923	0.12035			

3.2 result of test set:

结果部分截图如下：



	A	B	C	D	E	F	G	H
1	textid	Words (spl	anger	disgust	fear	joy	sad	surprise
2	1	who is prir	0.1220277	0.1771505	0.0436823	0.1656851	0.3086643	0.1827723
3	2	prestige hc	0	0	0.0208914	0.4146083	0.1700547	0.3944873
4	3	study femc	0.0234894	0.0304581	0.0568774	0.4235892	0.0923829	0.373203
5	4	no e book	0.0939017	0.0894851	0.0375257	0.2974184	0.206079	0.2756275
6	5	blair apolc	0.0992454	0.0412765	0.2686148	0	0.4829321	0.1079123
7	6	vegetables	4.91E-07	0	0.0926015	0.4443948	3.62E-06	0.4629996
8	7	afghan for	0.1002038	0.0087496	0.407718	0.0087496	0.4394656	0.0351134
9	8	skip the sh	0	0.1138773	0	0.1374779	0	0.7486675
10	9	made in ch	0.1056088	0.1277586	0.1886351	0.2227743	0.1001795	0.2550248
11	10	britain to s	0.138151	0.0369475	0.3090355	0.1550375	0.2155016	0.145285
12	11	canadian b	0.0575884	0	0.1168866	0.3184908	0.1104617	0.3965538
13	12	russia to st	0.1213503	0.0709482	0.3171112	0.1504688	0.1887585	0.151363
14	13	alzheimer	0.1325191	0.0659001	0.2038966	0.2188359	0.2221247	0.1567272
15	14	uk police s	0.167043	0.1359827	0.4160671	0.0190634	0.2134831	0.0483198

五、改进和结果分析

5.1 计算各种距离时，用上向量化的形式计算距离矩阵，提高效率

如欧式距离向量化推导：

$$|X-Y|^2=\sum(x_i-y_i)^2=\sum {x_i}^2+\sum {y_i}^2-2\sum x_i*y_i=|X|^2+|Y|^2-X*Y^T$$

5.2 为什么不同的K会出现不同的表现

个人理解是：过小的K会被training的样本的噪声干扰，会被一些非正常的偏移值引导，增加K可以增加抗干扰能力；过大的K会导致在投票选取最终label的时，所有测试数据都出现一样的结果，因为K过大，会使得距离较远的样本也参与了无效的"投票"

5.3 为什么不同距离函数会出现不同表现

原因其实在距离函数分析那里已经有所说明，不同距离函数青睐的特征不同，从而根据不同数据的特征向量的特点选择不同的距离函数。

六、思考题

1. IDF计算公式中分母为什么需要+1?

$$idf_j=log\frac{N}{1+\sum_{j=1}^{N}v_j},\ v_j=1\ if\ word_j\ appeared\ in\ text_n$$

但分母中的词频可能会为0，如果没有这个1的偏移的话，会出现除以一个0的计算错误；事实上，在处理TF-IDF矩阵的时候，我也用到了这个偏移的办法消除0值。

2.IDF数值有什么含义，TF-IDF数值有什么含义?

IDF可能可以作为一个词是否代表特殊语意的标准，**IDF值越小**说明这个词在所有不同文档出现的次数较多，越可能**不**传达有效信息（比如说the、I等通用名词），从这个角度看，IDF数值可以代替一个词对文档含义影响程度的权重值。

3.为什么KNN距离的倒数作为权重？如何使同一个测试样本的各个情感概率总和为1?

设：k是$label$的种数，n是测试样本个数

$$we\ have\ \sum_{j=1}^{k} p_{ij} = 1,$$

$$then,\ \sum_{j=1}^{k}\sum_{i=1}^{n} \frac{p_{ij}}{d_i} = \sum_{i=1}^{n}\sum_{j=1}^{k} \frac{p_{ij}}{d_i} = \sum_{i=1}^{n} \frac{1}{d_i}$$

所以我们可以在除以该放缩因子，使得每个样本的预测概率估计和为1