pkagglehw01

1自行寻找资料，了解**hyperopt**库的基本用法，尝试编写一个小例子

一段使用感知器判别鸢尾花数据的代码，使用的学习率是0.1,迭代40次得到了一个测试集上正确率为82%的结果。使用hyperopt优化参数，将正确率提升到了91%。

```
from sklearn import datasets
import numpy as np
from sklearn.cross_validation import train_test_split
from sklearn.metrics import accuracy_score
iris = datasets.load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)



from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

from sklearn.linear_model import Perceptron
ppn = Perceptron(n_iter=40, eta0=0.1, random_state=0)
ppn.fit(X_train_std, y_train)

y_pred = ppn.predict(X_test_std)
print (accuracy_score(y_test, y_pred))

def percept(args):
    global X_train_std,y_train,y_test
    ppn = Perceptron(n_iter=int(args["n_iter"]),eta0=args["eta"]*0.01,random_state=0)
    ppn.fit(X_train_std, y_train)
    y_pred = ppn.predict(X_test_std)
    return -accuracy_score(y_test, y_pred)

from hyperopt import fmin,tpe,hp,partial
space = {"n_iter":hp.choice("n_iter",range(30,50)),
         "eta":hp.uniform("eta",0.05,0.5)}
algo = partial(tpe.suggest,n_startup_jobs=10)
best = fmin(percept,space,algo = algo,max_evals=100)
print (best)
print (percept(best))
#0.822222222222
#{'n_iter': 14, 'eta': 0.12877033763511717}
#-0.911111111111
```

2Quora(国外版的知乎**)**每天都有成千上万的人在上面进行各个方面问题的提问，那么当中肯定有很多重复的、类似的提问，如何判别这些提问是否问的同一个问题？

（**1**）进行基本的数据探索，对数据的基本情况形式描述说明

（**2**）尝试从中提取一系列的有效特征，帮助解决该问题。

```
import pandas as pd
df = pd.read_csv('../data/train.csv')
all_questions = set(pd.unique(df['qid1'])) | set(pd.unique(df['qid2']))
print('问题总数：{}个'.format(len(all_questions)))
问题总数：537933个
#问题编码字典
q_dict = dict()
for line in df.itertuples():
    if line[2] not in q_dict:
        q_dict[line[2]] = line[4]
    if line[3] not in q_dict:
        q_dict[line[3]] = line[5]
q_similar_dict = dict()
index, count = 0, 0#统计最大的重复问题的字典索引和个数
for line in df.itertuples():
    if line[6] == 1:
        flag = 0
        for k,v in q_similar_dict.items():
            if line[2] in v or line[3] in v:
                v.add(line[2])
                v.add(line[3])
                if len(v)>count:
                    count = len(v)
                    index = k
                flag = 1
                break
        if flag == 0:
            q_similar_dict[len(q_similar_dict)] = set()
            q_similar_dict[len(q_similar_dict)-1].add(line[2])
            q_similar_dict[len(q_similar_dict)-1].add(line[3])

print('一共有{}组重复的问题'.format(len(q_similar_dict)))
一共有64026组重复的问题
```

3对于上述问题，你觉得可以怎么解决问题？简述你的思路

什么问题可能是相似的？ * 如果两个问题word重复的比例越高，越可能重复。 但是也不尽然。。。1)关键词很重要！2)还要考虑同义词。3)句式

把解决问题重复的问题 转化为 解决句子相似度 的问题。

输入两个句子，输出它们的相似度。

思路是避开特征提取的过程，用lstm来做。