part1 电影评价分类 苏潇 PB14011003

环境和依赖的库

OS:Windows10

Python version:2.7.13

机器学习相关算法主要是调用sklearn库

word2vec调用gensim库

另外稀疏矩阵依赖scipy

运行说明

dataset文件是SVD降维测试所用的数据子集.所有的代码均包含在main.py中

对于word2vec的测试在test子文件中,其中有1000条正面评价原文和1000条负面评价原文.

word2vec_getmodel.py读取所有的原文,进行word2vec训练,得到model,保存在my_real_model文件中. main.py直接从my_real_model中读取模型,然后根据模型获取词向量,进行交叉验证.

注:因为直接import了所有sklearn,运行时sklearn会报一些奇怪的warning,尝试了一下去除不掉,但不影响运行,可以不管. gensim在import时也会报一个warning信息,似乎是Windows的锅,也可以不管.

基本思想

特征提取

首先是读取给定的bag of words格式的数据,即DATA文件中的数据,然后把其转化成对应的向量,vocabulary中每个词对应向量的一维,其值为出现的次数.这是最原始的数据表示方式,但是缺点是由于词典比较大,有将近9万个词,所以维数会比较高.

这里采取了2种优化方法尝试,一种是直接使用sklearn的降维模块的SVD方法进行降维,可以指定降维的维数.实验中将其降到了100维,发现效果还是不错的.

另一种方法的想法是用word2vec,依赖gensim库.这样子需要用到原文,其好处在于原理是利用了深度学习原理,能够考虑到上下文之间的联系.这样需要进行一些简单的分词.

具体做法是:先把所有原文文件读入,分词后转换为word2vec要求的句子序列的输入形式,然后进行训练,就会得到各个词的词向量.默认100维.然后再遍历一遍原文,对每条评价,把其中出现过的词对应的词向量累加起来,再除以词的总数,作为这条comment的向量(即加权平均).这样就得到了所需的特征数据.

但实际实验中,因为时间仓促和内存原因,只利用1000条正面评价和1000条负面评价的子集进行word2vec训练.之后交叉验证时发现结果准确率并不太理想,最好的结果也只在60%-70%,与之前单纯的SVD降维反而效果要差,不知道是不是因为没有采用全部数据的原因. 当然自己对word2vec不是十分了解,也可能是参数设置不合适导致的.

朴素贝叶斯

朴素贝叶斯模型很简单,这里只需要直接调用sklearn库的相关API即可,这里调用的是多项式模型的朴素贝叶斯库.注意不能直接使用predict函数,而是要fit后根据proba获取预测的概率,再看其是否高于threshold来判断

最小二乘

最小二乘模型也比较简单,查阅了sklearn库的手册,发现可以利用Ridgeclassifier来实现.只要训练完毕后用predict预测,再统计一下正确率即可.

soft svm

这里同样也是利用sklearn库,sigma时直接用LinearSVC,其余用SVC,注意gamma与sigma的换算关系是:

gamma=(1.0/sigma**2) 即可.训练完毕后用predict预测,再统计一下正确率.

交叉验证

sklearn同样也提供了K-Fold交叉验证的模块,并提供了用例.利用给出的API,可以很方便完成数据的划分,只要按照要求的数据格式,进行计算,记录到矩阵里再返回即可.

注:这里SVM的相关参数d的计算,pdf里似乎没有说清是只针对划分的训练集还是针对全部的数据集.这里为了方便,就直接对全部数据集进行计算了,不然还要每次划分后再计算一次,速度会更慢.因为是训练集:测试集=4:1,效果应该不会有太大差别

实验测试结果

SVD降维

因为原始数据太大,跑起来速度实在太慢,所以从里面随机选取了1000条正例和1000条负例做结果测试,首先是采用SVD降维的结果,如下:

naive bayes:

```
[[ 0.86613387  0.87412587  0.882       0.864       0.861     ]
 [ 0.86013986  0.87112887  0.879       0.861       0.86      ]
 [ 0.85914086  0.87112887  0.88        0.86        0.856     ]
 [ 0.85714286  0.87112887  0.88        0.861       0.856     ]
 [ 0.85314685  0.87512488  0.879       0.859       0.854     ]
 [ 0.85114885  0.86913087  0.876       0.861       0.854     ]
 [ 0.84915085  0.86513487  0.873       0.855       0.853     ]]
```

平均准确率:

[0.86945195 0.86625375 0.86525395 0.86505435 0.86405435 0.86225594 0.85905714]

可见基本可以达到85%以上,最大约87%,准确率比较理想。当然这也和因为输入的数据是原始未降维过的数据,所以信息比较完整有关系。

最小二乘法:

```
[[ 0.76623377  0.77922078  0.794       0.78        0.793     ]
 [ 0.76623377  0.77922078  0.794       0.78        0.793     ]
 [ 0.76623377  0.77922078  0.794       0.78        0.793     ]
 [ 0.76623377  0.77922078  0.794       0.78        0.793     ]
 [ 0.76623377  0.77822178  0.794       0.78        0.793     ]
 [ 0.76623377  0.77822178  0.794       0.78        0.793     ]
 [ 0.76623377  0.77822178  0.795       0.78        0.793     ]
 [ 0.76823177  0.77822178  0.793       0.781       0.795     ]
 [ 0.76723277  0.78121878  0.793       0.779       0.792     ]
 [ 0.75524476  0.76823177  0.765       0.771       0.781     ]
 [ 0.74925075  0.74525475  0.755       0.751       0.76      ]]
```

平均准确率:

[0.78249091 0.78249091 0.78249091 0.78249091 0.78229111 0.78229111
  0.78249111 0.78309071 0.78249031 0.7680953   0.7521011]

最大值约为78%,可见最小二乘法的参数选取对结果影响不是很大,多数结果平均准确率都在75%-78%左右,效果还比较理想

SVM:

```
[[ 0.71428571  0.74225774  0.739       0.719       0.739     ]
 [ 0.6993007   0.74825175  0.73        0.724       0.72      ]
 [ 0.68931069  0.73026973  0.727       0.72        0.709     ]
 [ 0.68931069  0.73126873  0.727       0.72        0.709     ]
 [ 0.74625375  0.70729271  0.73        0.719       0.718     ]
 [ 0.76523477  0.79120879  0.77        0.774       0.781     ]
 [ 0.76423576  0.77522478  0.787       0.776       0.799     ]
 [ 0.75724276  0.76823177  0.787       0.779       0.782     ]
 [ 0.49350649  0.48151848  0.486       0.496       0.485     ]
 [ 0.53246753  0.51348651  0.562       0.523       0.639     ]
 [ 0.74925075  0.71228771  0.734       0.731       0.723     ]
 [ 0.76023976  0.78421578  0.778       0.776       0.782     ]
 [ 0.49350649  0.48151848  0.486       0.496       0.485     ]
 [ 0.49350649  0.48151848  0.486       0.496       0.485     ]
 [ 0.49350649  0.48151848  0.486       0.496       0.485     ]
 [ 0.53246753  0.51348651  0.562       0.522       0.644     ]
 [ 0.49350649  0.48151848  0.486       0.496       0.485     ]
 [ 0.49350649  0.48151848  0.486       0.496       0.485     ]
 [ 0.49350649  0.48151848  0.486       0.496       0.485     ]
 [ 0.49350649  0.48151848  0.486       0.496       0.485     ]]
```

平均准确率:

[0.73070869 0.72431049 0.71511608 0.71531588 0.72410929 0.77628871
  0.78029211 0.77469491 0.488405   0.55399081 0.72990769 0.77609111
  0.488405   0.488405   0.488405   0.55479081 0.488405   0.488405
  0.488405   0.488405]

可见SVM的效果波动较大,与参数的选取有较大关系,最大值大概78%,效果也还是不错的,但有的参数下也会低到48%.

最佳参数选取如下:

bayes best parm index: 0
bayes best parm: 0.5
ls best parm index 7
ls best parm: 100
svm best parm index 1 2
svm best parm: 0.100000*d 100

Word2Vec

然后是采用原文,word2vec训练,原始数据是前1000条正面评价和最后1000条负面评价,这里因为贝叶斯分类器不能使用线性数据,所以只对最小二乘和SVM验证.

结果如下:

ls res:

[[0.74193548 0.73383085 0.74378109 0.73134328 0.75124378]
 [0.68238213 0.68656716 0.71144279 0.69402985 0.71393035]
 [0.66004963 0.65174129 0.6840796   0.65920398 0.69154229]
 [0.65012407 0.62935323 0.64676617 0.64925373 0.69900498]
 [0.63771712 0.61940299 0.62935323 0.64427861 0.70149254]
 [0.62282878 0.6119403   0.60945274 0.64179104 0.7039801]
 [0.5955335   0.59452736 0.61442786 0.63432836 0.68656716]
 [0.56327543 0.56467662 0.58457711 0.58955224 0.65422886]
 [0.56327543 0.53482587 0.56965174 0.539801   0.62189055]
 [0.56079404 0.49751244 0.49253731 0.49004975 0.539801]
 [0.57320099 0.48507463 0.460199   0.46766169 0.49502488]]
ls avg:

[0.7404269   0.69767046 0.66932336 0.65490044 0.6464489   0.63799859
  0.62507685 0.59126205 0.56588892 0.51613891 0.49623224]

可见效果波动较大,最好大概有74%,与前者相比稍差.

svm res:

[[0.50372208 0.47263682 0.46268657 0.46268657 0.48507463]
 [0.50372208 0.47263682 0.46268657 0.46268657 0.48507463]
 [0.50372208 0.47263682 0.46268657 0.46268657 0.48507463]
 [0.50372208 0.47263682 0.46268657 0.46268657 0.48507463]
 [0.51116625 0.53731343 0.49751244 0.47014925 0.52238806]
 [0.50868486 0.54726368 0.50746269 0.4800995   0.52487562]
 [0.50868486 0.54726368 0.50746269 0.4800995   0.52487562]
 [0.50868486 0.54726368 0.50746269 0.4800995   0.52487562]
 [0.63771712 0.62686567 0.59701493 0.64427861 0.65174129]
 [0.66253102 0.64925373 0.63432836 0.63681592 0.65671642]
 [0.65260546 0.61442786 0.64179104 0.6641791   0.64676617]
 [0.63027295 0.60199005 0.60447761 0.67661692 0.63930348]
 [0.55831266 0.54726368 0.58457711 0.56965174 0.6318408]
 [0.60794045 0.59950249 0.59950249 0.62686567 0.65920398]
 [0.63275434 0.61442786 0.62686567 0.65174129 0.71144279]
 [0.66997519 0.64925373 0.65422886 0.66666667 0.67661692]

[0.49875931 0.47263682 0.460199   0.45771144 0.48258706]
[0.49875931 0.47263682 0.460199   0.45771144 0.48258706]
[0.55334988 0.54726368 0.57960199 0.56716418 0.6318408]
[0.59305211 0.57960199 0.60945274 0.62189055 0.65671642]]

svm avg:

[0.47736133 0.47736133 0.47736133 0.47736133 0.50770589 0.51367727
  0.51367727 0.51367727 0.63152352 0.64792909 0.64395393 0.6305322
  0.5783292   0.61860301 0.64744639 0.66334827 0.47437873 0.47437873
  0.5758441   0.61214276]

最大值也只有66%,而且很多情况下只在50%左右徘徊,可见效果并不是很理想.

最佳参数选择如下:

ls best parm index: 0
ls best parm: 0.0001
svm best parm index: 3, 3
svm best parm: 10.000000*d 1000

实验总结

通过本次实验,完整地接触了应用多种监督学习经典算法的过程,也了解了sklearn这一机器学习的常用库.同时也试着去用和深度学习,神经网络等有一定联系的word2vec来进行了特征提取,但是由于自己对这方面了解较少,加上时间仓促,最后使用word2vec的效果不是太好.反而简单的SVD降维还是有着不错的效果.但这也是一次有意义的尝试.

三种模型横向对比,仅就我自己的实验结果来说,使用最原始的数据输入的朴素贝叶斯模型虽然简单,但取得的效果是最好的.最小二乘分类器和SVM的最好情况准确率基本相近,但SVM的效果随参数选取的波动相对来说会大一些,需要在选择参数时更加小心.