大数据竞赛平台——Kaggle 入门篇

这篇文章适合那些刚接触Kaggle、想尽快熟悉Kaggle并且独立完成一个竞赛项目的网友,对于已经在Kaggle上参赛过的网友来说,大可不必耗费时间阅读本文。本文分为两部分介绍Kaggle,第一部分简单介绍Kaggle,第二部分将展示解决一个竞赛项目的全过程。如有错误,请指正!

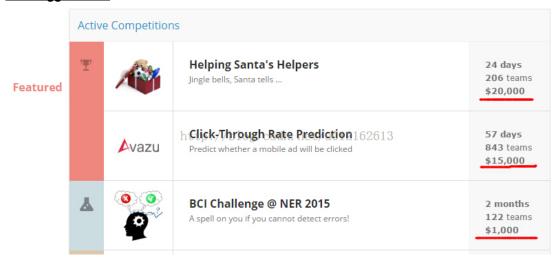
1、Kaggle简介

Kaggle是一个数据分析的竞赛平台,网址: https://www.kaggle.com/

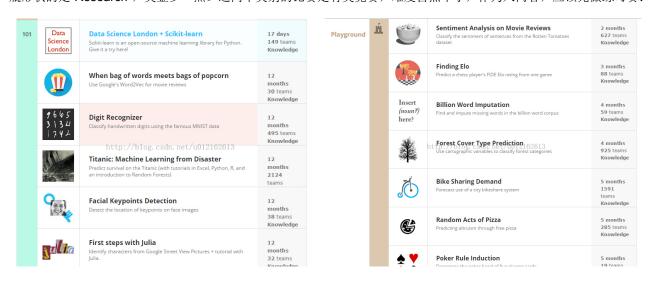
企业或者研究者可以将数据、问题描述、期望的指标发布到Kaggle上,以竞赛的形式向广大的数据科学家征集解决方案,类似于KDD-CUP(国际知识发现和数据挖掘竞赛)。Kaggle上的参赛者将数据下载下来,分析数据,然后运用机器学习、数据挖掘等知识,建立算法模型,解决问题得出结果,最后将结果提交,如果提交的结果符合指标要求并且在参赛者中排名第一,将获得比赛丰厚的奖金。更多内容可以参阅:大数据众包平台

下面我以图文的形式介绍Kaggle:

进入Kaggle网站:

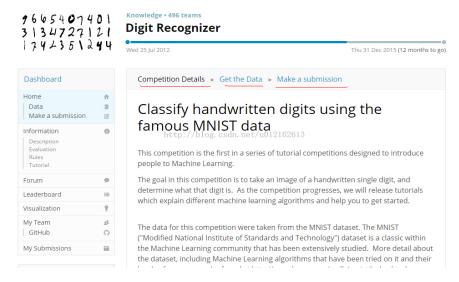


这是当前正在火热进行的有奖比赛,有冠军杯形状的是"Featured",译为"号召",召集数据科学高手去参赛。下面那个灰色的有试剂 瓶形状的是"Research",奖金少一点。这两个类别的比赛是有奖竞赛,难度自然不小,作为入门者,应该先做练习赛:

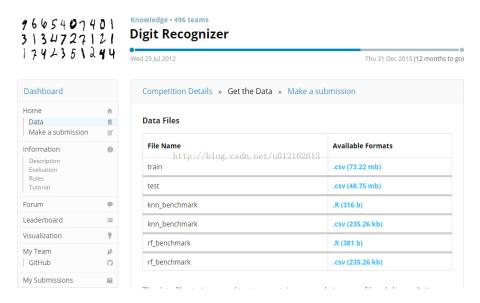


左图的比赛是"101",右图的是"Playground",都是练习赛,适合入门。入门Kaggle最好的方法就是独立完成101和playground这两个级别的竞赛项目。本文的第二部分将选101中的"Digit Recognition"作为讲解。

点击进入赛题"Digit Recognition ::

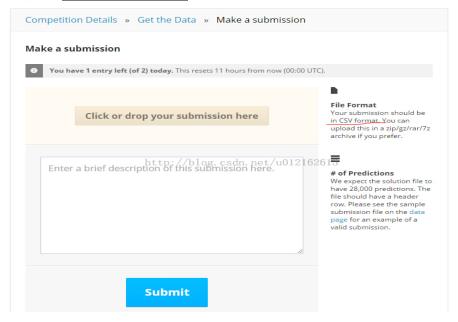


这是一个识别数字0~9的练习赛,"Competition Details"是这个比赛的描述,说明参赛者需要解决的问题。"Get the Data"是数据下载,参赛者用这些数据来训练自己的模型,得出结果,数据一般都是以csv格式给出:



其中,train.csv就是训练样本,test.csv就是测试样本,由于这个是训练赛,所以还提供了两种解决方案,knn_benchmark.R和 rf_benchmark.R,前者是用R语。言写的knn算法程序,后者是用R语言写的随机森林算法程序,它们的结果分别是 knn benchmark.csv和rf benchmark.csv。关于csv格式文件,我前一篇文章有详述:【Python】csv模块的使用。

得出结果后,接下来就是提交结果"Make a submission":



要求提交的文件是csv格式的,假如你将结果保存在result.csv,那么点击"Click or drop submission here",选中result.csv文件上传即可,系统将测试你提交的结果的准确率,然后排名。

另外,除了"Competition Details"、"Get the Data"、"Make a submission",侧边栏的"Home"、"Information"、"Forum"等,也提供了关于竞赛的一些相关信息,包括排名、规则、辅导……

【以上是第一部分,暂且写这么多,有补充的以后再更】

2、竞赛项目解题全过程

(1) 知识准备

首先,想解决上面的题目,还是需要一点ML算法的基础的,另外就是要会用编程语言和相应的第三方库来实现算法,常用的有: Python以及对应的库numpy、scipy、scikit-learn(实现了ML的一些算法,可以直接用)、theano(DeepLearning的算法包)。 R语言、weka

如果用到深度学习的算法,cuda、caffe也可以用

总之,使用什么编程语言、什么平台、什么第三方库都无所谓,无论你用什么方法,Kaggle只需要你线上提交结果,线下你如何实现算法是没有限制的。

Ok,下面讲解题过程,以"Digit Recognition"为例,数字识别这个问题我之前写过两篇文章,分别用kNN算法和Logistic算法去实现,有完整的代码,有兴趣可以阅读:kNN算法实现数字识别、Logistic回归实现数字识别

(2) Digit Recognition解题过程

下面我将采用kNN算法来解决Kaggle上的这道Digit Recognition训练题。上面提到,我之前用kNN算法实现过,这里我将直接copy之前的算法的核心代码,核心代码是关于kNN算法的主体实现,我不再赘述,我把重点放在处理数据下载以下工程基于<u>Python、numpy</u>

• 获取数据

从"Get the Data"下载以下三个csv文件:

knn_benchmark.csv	2014/12/7 13:38	Microsoft Excel	236 KB
test.csv	2014/12/7 14:13 http://blog.csdp.net	Microsoft Excel	49,921 KB
train.csv	nttp://blogof471277 14:40	Microsoft Excel	74,976 KB

• 分析train.csv数据

train.csv是训练样本集,大小42001*785,第一行是文字描述,所以实际的样本数据大小是42000*785,其中第一列的每一个数字是它对应行的label,可以将第一列单独取出来,得到42000*1的向量trainLabel,剩下的就是42000*784的特征向量集trainData,所以从train.csv可以获取两个矩阵trainLabel、trainData。

下面给出代码,另外关于如何从csv文件中读取数据,参阅:csv模块的使用

```
[python] 🗎 📑
91.
      def loadTrainData():
02.
          1=[]
03.
          with open('train.csv') as file:
94
              lines=csv.reader(file)
               for line in lines:
05.
                  1.append(line) #42001*785
06.
07.
          1.remove(1[0])
08.
          l=array(1)
09.
          label=1[:,0]
          data=1[:,1:]
10.
          return nomalizing(toInt(data)),toInt(label)
11.
```

这里还有两个函数需要说明一下,toInt()函数,是将字符串转换为整数,因为从csv文件读取出来的,是字符串类型的,比如 '253',而我们接下来运算需要的是整数类型的,因此要转换,int('253')=253。toInt()函数如下:

```
[python] 🖥 📋
01.
      def toInt(array):
02.
          array=mat(array)
03.
          m,n=shape(array)
04.
          newArray=zeros((m,n))
05.
          for i in xrange(m):
06.
          for j in xrange(n):
07.
                      newArray[i,j]=int(array[i,j])
08.
          return newArray
```

nomalizing()函数做的工作是归一化,因为train.csv里面提供的表示图像的数据是0~255的,为了简化运算,我们可以将其转化为二值图像,因此将所有非0的数字,即1~255都归一化为1。nomalizing()函数如下:

```
[python] i i def nomalizing(array):
```

· 分析test.csv数据

test.csv里的数据大小是28001*784,第一行是文字描述,因此实际的测试数据样本是28000*784,与train.csv不同,没有label,28000*784即28000个测试样本,我们要做的工作就是为这28000个测试样本找出正确的label。<u>所以从test.csv我们可以得到测试样本集testData</u>,代码如下:

```
[python] 🖥 📋
01.
      def loadTestData():
02.
          1=[]
03.
          with open('test.csv') as file:
04.
             lines=csv.reader(file)
05.
               for line in lines:
06.
                  1.append(line)
07.
08.
          1.remove(1[0])
99.
          data=array(1)
10.
          return nomalizing(toInt(data))
```

• 分析knn_benchmark.csv

前面已经提到,由于digit recognition是训练赛,所以这个文件是官方给出的参考结果,本来可以不理这个文件的,但是我下面为了对比自己的训练结果,所以也把knn_benchmark.csv这个文件读取出来,这个文件里的数据是28001*2,第一行是文字说明,可以去掉,第一列表示图片序号1~28000,第二列是图片对应的数字。从knn_benchmark.csv可以得到28000*1的测试结果矩阵testResult,代码:

```
[python] 🗎 📋
01.
      def loadTestResult():
02.
          1=[]
          with open('knn_benchmark.csv') as file:
03.
              lines=csv.reader(file)
94
05.
               for line in lines:
06.
                  1.append(line)
07.
           #28001*2
08.
          1.remove(1[0])
09.
          label=array(1)
          return toInt(label[:,1])
10.
```

到这里,数据分析和处理已经完成,我们获得的矩阵有: trainData、trainLabel、testData、testResult

• 算法设计

这里我们采用kNN算法来分类,核心代码:

```
[python] 🖥 📋
01.
      def classify(inX, dataSet, labels, k):
92.
          inX=mat(inX)
03.
          dataSet=mat(dataSet)
04.
          labels=mat(labels)
05.
          dataSetSize = dataSet.shape[0]
06.
          diffMat = tile(inX, (dataSetSize,1)) - dataSet
07.
          sqDiffMat = array(diffMat)**2
          sqDistances = sqDiffMat.sum(axis=1)
08.
09.
          distances = sqDistances**0.5
10.
          sortedDistIndicies = distances.argsort()
11.
          classCount={}
12.
          for i in range(k):
13.
              voteIlabel = labels[0,sortedDistIndicies[i]]
14.
              classCount[voteIlabel] = classCount.get(voteIlabel,0) + 1
15.
          sortedClassCount = sorted(classCount.iteritems(), key=operator.itemgetter(1), reverse=True)
16.
          return sortedClassCount[0][0]
```

关于这个函数,参考: kNN算法实现数字识别

简单说明一下,inX就是输入的单个样本,是一个特征向量。dataSet是训练样本,对应上面的trainData,labels对应trainLabel,k 是knn算法选定的k,一般选择0~20之间的数字。这个函数将返回inX的label,即图片inX对应的数字。 对于测试集里28000个样本,调用28000次这个函数即可。

• 保存结果

kaggle上要求提交的文件格式是csv,上面我们得到了28000个测试样本的label,必须将其保存成csv格式文件才可以提交,关于csv,参考: 【Python】csv模块的使用。 代码:

```
python] def saveResult(result):
    with open('result.csv','wb') as myFile:
        myWriter=csv.writer(myFile)
    for i in result:
        tmp=[]
        tmp.append(i)
        myWriter.writerow(tmp)
```

• 综合各函数

上面各个函数已经做完了所有需要做的工作,现在需要写一个函数将它们组合起来解决digit recognition这个题目。我们写一个handwritingClassTest函数,运行这个函数,就可以得到训练结果result.csv。

```
[python] 🖥 📋
01.
      def handwritingClassTest():
02.
          trainData,trainLabel=loadTrainData()
03.
          testData=loadTestData()
04.
          testLabel=loadTestResult()
95.
          m.n=shape(testData)
06.
          errorCount=0
07.
          resultList=[]
          for i in range(m):
08.
09.
               classifierResult = classify(testData[i], trainData, trainLabel, 5)
              resultList.append(classifierResult)
10.
               print "the classifier came back with: %d, the real answer is: %d" % (classifierResult, testLabel[0,i])
11.
12.
              if (classifierResult != testLabel[0,i]): errorCount += 1.0
13.
          print "\nthe total number of errors is: %d" % errorCount
14.
          print "\nthe total error rate is: %f" % (errorCount/float(m))
15.
          saveResult(resultList)
```

色1101四数,	uit.csv X IT:	
<u>-</u>		→ D C
A1	*	\mathbb{R} fx

	A1	*	R	fx
	A	В		
1	2			
2	2 0 9 3 7 0 3			
3	9			
4	9			
2 3 4 5 6 7 8	3			
6	7			
7	0			
8	3			
9	0			
10	3 5			
11	5			
12	7			
13	4			
14	0			
15	4 5			
16				
17	3			
18	1			
19	9			
20	0			
21	9			
22	1			
23	1			
24	5			
25	7			
26	4			
27	2			

```
🦆 Python 1 🗵
the classifier came back with: 2, the real answer is: 2
the classifier came back with: 2, the real answer is: 2
the classifier came back with: 9, the real answer is: 9
the classifier came back with: 6, the real answer is: 6
the classifier came back with: 7, the real answer is: 7
the classifier came back with: 6, the real answer is: 6
the classifier came back with: 1, the real answer is: 1
the classifier came back with: 9, the real answer is: 9
the classifier came/back with: (7), the [réal] answer is: 9 the classifier came back with: 9, the real answer is: 9
the classifier came back with: 7, the real answer is: 7
the classifier came back with: 3, the real answer is: 3
the classifier came back with: 9, the real answer is: 9
the classifier came back with: 2, the real answer is: 2
the total number of errors is: 1004
the total error rate is: 0.035857
```

28000个样本中有1004个与kknn_benchmark.csv中的不一样。错误率为3.5%,这个效果并不好,原因是我并未将所有训练样本都拿来训练,因为太花时间,我只取一半的训练样本来训练,即上面的结果对应的代码是:

训练一半的样本,程序跑了将近70分钟(在个人PC上)。

• 提交结果

将result.csv整理成kknn_benchmark.csv那种格式,即加入第一行文字说明,加入第一列的图片序号,然后make a submission,结果准确率96.5%:



下载工程代码: github地址

【完】