如何从图片文件中还原出统计数据

计算士 Feb.2, 2012

我们常常会接触到一些研究报告，这些报告里没有原始数据，只有统计图，例如：

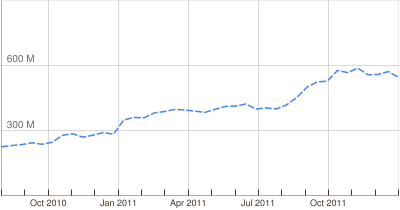


图1. Facebook在一年半内的独立用户访问量变化

这张图是通过https://www.google.com/adplanner/#siteSearch查询到的Facebook在全球范围内的流量变化。从图上可以看出，从2010年8月1日到2012年12月31日，该网站的独立用户（cookie）访问量从2亿多升到了将近6亿。

如果我们仅仅得到以上结论的话，这张图的信息利用率是极低的。我们再仔细观察一下这张图。x轴上每一小格是一个月，在小格内曲线有变动，说明生成该图的原始数据是以周或天为单位的。姑且让我们假设原数据是以天为单位变动的，那我们能否从该图中还原出原数据呢？

答案是肯定的。我们可以采取人工的方法，一点一点地丈量蓝色曲线的变化，把坐标记录下来，这就是“伪”原始数据。当然，我们也可以把这个过程交给程序处理。

我设想了一种使用计算机程序从曲线中还原数据的方法，分为如下步骤：

1. 切图：将原始图像的坐标轴外围的标记切去；
2. 读图：将切后的图读为一个像素矩阵；
3. 分离曲线：把曲线从背景中分离出来；
4. 计算坐标：计算曲线上的像素点在像素矩阵中的位置；
5. 还原数据：将像素坐标转化为真实数据。

使用Mathematica语言，我将这些步骤逐一实现，并打包成一个函数叫retrieveCurveData，它可以读入一张图并还原出曲线背后隐藏的原始数据。下面，我将以图一为案例，逐一讲解这些步骤：

1. 切图

因为我们要把点的像素坐标转化为原始数据坐标，所以必须要把一张图多余的部分，使得其上的点在像素矩阵里的坐标与坐标轴位置一致。在本案例中，我们只需要把x轴的坐标切去即可。

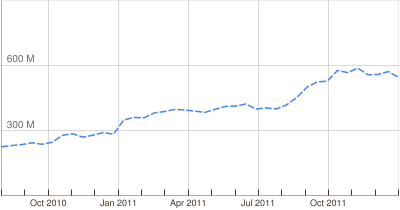
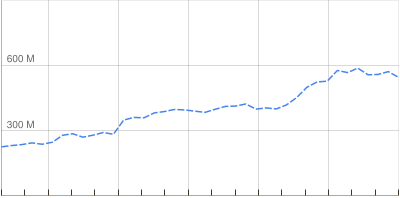
 

图2. （左）切图前；（右）切图后

1. 读图

我们可以将一张图读成一个像素矩阵。本例中，切前的图是210 × 400的矩阵，即有210行像素，400列像素，切后是198 × 400。其中，每一个像素可被表达为RGBColor，即红绿蓝三色标识。所有的计算机屏幕显示的颜色都可以通过混合红绿蓝三种颜色得到，一些典型颜色的标识如下表所示。

表1. 典型颜色的三色标识

|  |  |  |  |
| --- | --- | --- | --- |
|  | Red | Green | Blue |
|  | 1 | 0 | 0 |
|  | 0 | 1 | 0 |
|  | 0 | 0 | 1 |
|  | 0.1 | 0.6 | 0.9 |
|  | 1 | 1 | 1 |

1. 分离曲线

如前所诉，我们得到了一堆像素点，现在我们的任务是把曲线上的像素点提取出来。这个任务看起来有点难，其实解决起来很简单。

从计算机科学的角度，我们可以把三种颜色看做是一个三维空间，每个像素点都位于这个三维空间中。我们的任务就是把点在三维空间中分组。从社会科学的角度，本问题的情境也是我们熟悉的。我们可以把这些像素点看作是社会调查数据中的被访者，每个被访者在红色、蓝色、绿色上的取值是他们的人口统计学变量取值（例如年龄、收入与教育程度），这时我们自然而然就会想到用聚类分析把这些“人”分成不同的组。

经过几次简单的调试，我发现分成5组时效果比较好。抽取各组的“成员”显示如下：

表2. 分组的结果

|  |  |
| --- | --- |
| Groups | Members |
| 1 | test.png |
| 2 | test.png |
| 3 | test.png |
| 4 | test.png |
| 5 | test.png |

每一组内的圆形标识，看起来颜色差不多，但其实它们的三色坐标有着细微的区别。我们选择第三组作为代表曲线的像素点。虽然第一组和第二组也是蓝色，但深蓝的像素点位于曲线“正中间”，而浅蓝的像素点位于曲线与白色背景的交叉地带，因此使用深蓝色的像素比较合适。

虽然像素矩阵有198 × 400 = 72900个点，但因为大多数点都是白色的，最后实际需要进行分组的点只有1141个，因此上述聚类分析的计算速度是很快的。

1. 计算坐标

现在我们要像素矩阵里去，匹配出被列入第三组的像素点的位置。得到它们的位置{x,y}后，把x × A / a就完成了从像素坐标到原始坐标的变换。其中a是像素矩阵列数，本例中等于400; A是原始数据中的总天数，本例中即2010年8月1日至2012年12月31日之间的天数，为518。对y轴也做类似操作 y × B / b：像素矩阵有198行，实际数据的最大值是9亿。

1. 还原数据

经过上述操作，我们得到287个数据点（由于原图的数据不是连续的曲线，而是有间断的虚线，再加上在聚类分析时排除了一些蓝色像素点，所以我们只能还原出一部分数据）。将得到的数据画出来和原图对比如下：

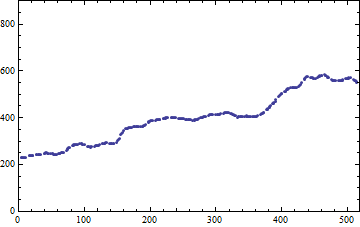
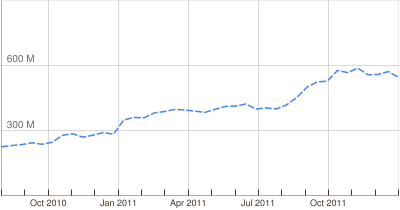


图3. （左）原图；（右）还原数据画出的新图

对比两个图，可以看到它们的相似性极高。为了保证这种做法的科学性，我们需要进行系统性的检验。

检验的方法如下：我们先取得一份真实数据，画出一张图，使用以上方法还原出数据，对比还原出的数据与真实数据之间的差异。

我们在互联网上获取2010年8月1日到2012年2月1日的道琼斯指数，因为周六周日不开盘，所以一共有380个数据点。

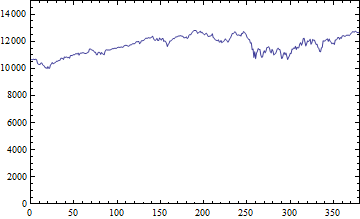
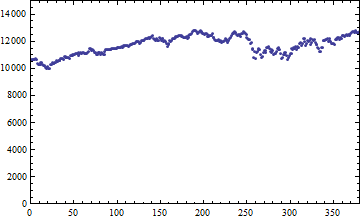


图4. 道琼斯指数变化的离散图（左）和曲线图（右）

正如互联网上公布数据时常有的做法，我们把离散的数据连起来画成曲线。对图4（右）进行分析，我们还原出185个数据点，与原始数据画在同一张图中对比如下：

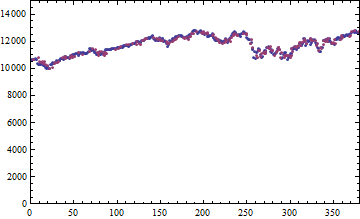


图4. 道琼斯指数原图（蓝色点）和恢复数据图（紫色点）

对比原始数据与恢复的数据，发现在x坐标重合（同一天）的数值点上，恢复数据的误差范围为原始数据的 ± 0.05，误差分布如下图。平均误差为0.01。

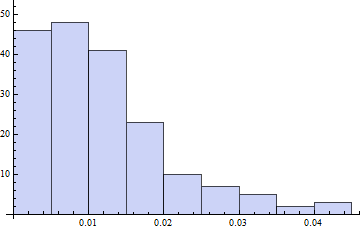


图4. 恢复数据与原始数据误差的分布

使用双样本Kolmogorov-Smirnov test得到的P值为1，说明不能拒绝这两组数据来源于同一个分布的假设（更严格地说，这两组数据来自不同分布的概率几乎是0）。

最后，我们以对本程序的局限和应用范围讨论结束本文。从以上分析中看出，本程序只适用于曲线与背景和坐标颜色不同的统计图。如果曲线和坐标都是黑色的，就不能用这个程序（需要考虑采取更复杂的图像识别的方法）。同时，本程序依赖于曲线的形状和格式，如果曲线比较细，而且是连续曲线，我们就能保留较多的信息。由于本程序的Reliability依赖于待分析的对象，因此较适合在手动修改各项参数后，用于分析一批同样格式的图片。

由于本文用到的方法没有参考其他文献，也尚未得到广泛认可，因此程序的使用者需自负风险。尤其值得一提的是，本程序也许可以用于获取一些纸版论文的数据（可以扫描后使用本程序恢复数据），为广大研究人员执行低成本的Meta-analysis提供了一个新的渠道。

附：retrieveCurveData程序的源代码

（\*Function: retrieveCurveData

Version: 1.0

Date: 2012-02-03

Author: Lingfei WU

Maintainer: Lingfei WU < wlf850927@gmail.com >

Depends: Mathematica (>= 7.0)

Description: Retrieve data of curves from images.

Date/Publication: 2012-02-02 00:04

Annotation: This Mathematica package is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. This Mathematica package is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. Please use at your own risk. See the GNU Library General Public License for more details. \*）

retrieveCurveData[figure\_, xMax\_, yMax\_] := Module[

{pdata, x, y, pdata1, pdata2, pdata3, positions1, positions2, positions3, positions4, days, getMeanTraffic, traffics, realTraffics, realDays, finaldata},

pdata = ImageData[figure];

{x, y} = ImageDimensions[figure];

pdata1 = Select[Flatten[pdata, 1], Length@Tally@# > 1 &];

pdata2 = FindClusters[pdata1, 5];

pdata3 = Union[pdata2[[3]]];

positions1 = Position[pdata, #] & /@ pdata3;

positions2 = Union[Reverse /@ Flatten[positions1, 1]];

positions3 = Transpose[{positions2[[All, 1]], y - positions2[[All, 2]]}];

positions4 = Transpose[{Round[positions3[[All, 1]]\*xMax/x],

positions3[[All, 2]]}];

days = Union[positions4[[All, 1]]];

getMeanTraffic[day\_] := Mean@Select[positions4, #[[1]] == day &][[All, 2]];

traffics = getMeanTraffic /@ days;

realTraffics = traffics\*yMax/y // N;

finaldata = Transpose[{days, realTraffics}]

]