« 异乡春节 ImageMagick Tips: Shadow your picture (again) »

漫谈 Clustering (番外篇): Vector Quantization

by pluskid, on 2009-01-13, in Machine Learning 25 comments

本文是"漫谈 Clustering 系列"中的第 3 篇, 参见本系列的其他文章。

在接下去说其他的聚类算法之前, 让我们先插进来说一说一个有点跑题的东西:Vector Quantization 。这项技术广泛地用在信号处理以及数据压缩等领域。事实上, 在 JPEG 和 MPEG-4 等多媒体压缩格式里都有 VQ 这一步。

Vector Quantization 这个名字听起来有些玄乎, 其实它本身并没有这么高深。大家都知道, 模拟信号是连续的值, 而计算机只能处理离散的数字信号, 在将模拟信号转换为数字信号的时候, 我们可以用区间内的某一个值去代替着一个区间, 比如, [0, 1) 上的所有值变为 0 , [1, 2) 上的所有值变成 1 , 如此类推。其这就是一个 VQ 的过程。一个比较正式一点的定义是:VQ 是将一个向量空间中的点用其中的一个有限子集来进行编码的过程。

一个典型的例子就是图像的编码。最简单的情况, 考虑一个灰度图片, 0 为黑色, 1 为白色, 每个像素的值为 [0, 1] 上的一个实数。现在要把它编码为 256 阶的灰阶图片, 一个最简单的做法就是将每一个像素值 x 映射为一个整数 `floor(x*255)` 。当然, 原始的数据空间也并不以一定要是连续的。比如, 你现在想要把压缩这个图片, 每个像素只使用 4 bit (而不是原来的 8 bit) 来存储, 因此, 要将原来的 [0, 255] 区间上的整数值用 [0, 15] 上的整数值来进行编码, 一个简单的映射方案是 `x*15/255` 。

不过这样的映射方案颇有些 Naive , 虽然能减少颜色数量起到压缩的效果, 但是如果原来的颜色并不是均匀分布的, 那么的出来的图片质量可能并不是很好。例如, 如果一个 256 阶灰阶图片完全由 0 和 13 两种颜色组成, 那么通过上面的映射就会得到一个全黑的图片, 因为两个颜色全都被映射到 0 了。一个更好的做法是结合聚类来选取代表性的点。

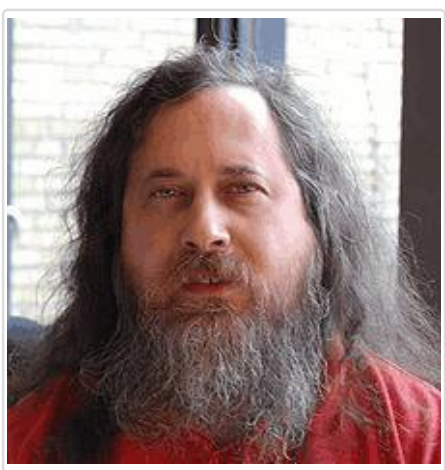
VQ 2

实际做法就是:将每个像素点当作一个数据, 跑一下 K-means , 得到 k 个 centroids , 然后用这些 centroids 的像素值来代替对应的 cluster 里的所有点的像素值。对于彩色图片来说, 也可以用同样的方法来做, 例如 RGB 三色的图片, 每一个像素被当作是一个 3 维向量空间中的点。

用本文开头那张 Rechard Stallman 大神的照片来做一下实验好了, VQ 2、VQ 10 和 VQ 100 三张图片分别显示聚类数目为 2 、10 和 100 时得到的结果, 可以看到 VQ 100 已经和原图非常接近了。把原来的许多颜色值用 centroids 代替之后, 总的颜色数量减少了, 重复的颜色增加了, 这种冗余正是压缩算法最喜欢的。考虑一种最简单的压缩办法:单独存储(比如 100 个)centroids 的颜色信息, 然后每个像素点存储 centroid 的索引而不是颜色信息值, 如果一个 RGB 颜色值需要 24 bits 来存放的话, 每个(128 以内的)索引值只需要 7 bits 来存放, 这样就起到了压缩的效果。

实现代码很简单, 直接使用了 SciPy 提供

VQ 10



VQ 100

的 `kmeans` 和 `vq` 函数, 图像读写用了 Python Image Library :

```
#!/usr/bin/python

from scipy.cluster.vq import kmeans, vq
from numpy import array, reshape, zeros
from mltk import image

vqclst = [2, 10, 100, 256]

data = image.read('example.jpg')
(height, width, channel) = data.shape

data = reshape(data, (height*width, channel))
for k in vqclst:
    print 'Generating vq-%d...' % k
    (centroids, distor) = kmeans(data, k)
    (code, distor) = vq(data, centroids)
    print 'distor: %.6f' % distor.sum()
    im_vq = centroids[code, :]
    image.write('result-%d.jpg' % k, reshape(im_vq,
        (height, width, channel)))
```

当然, Vector Quantization 并不一定要用 K-means 来做, 各种能用的聚类方法都可以用, 只是 K-means 通常是最简单的, 而且通常都够用了。

Tags: Clustering, Unsupervised Learning

25 comments to 漫谈 Clustering (番外篇): Vector Quantization