A Compressed Sense of Compressive Sensing (II)

http://freemind.pluskid.org/machine-learning/a-compressed-sense-of-compressive-sensing-ii

posted on Free Mind on August 14, 2013
generated with pandoc on December 3, 2015
category: Machine Learning

tags: Compressive Sensing, Fun

今天在吃饭的时候，其中一个人得知我的学校之后非要出一些 puzzle 来考我……口算都很困难的我觉得压力山大，要给本校的 CS 算法天才们丢脸了。然后其中一道题就是十二球重量的问题，似乎是一个经典面试题。当时想了一下觉得确实是可以放在之前看的一些 Compressive Sensing 框架下来考虑这个问题，后来去查了一下，发现果真如此。

Figure 1: .....

十二球问题是说，现在有十二个球，其中有一个是“异类”，除了这个异类之外，其他十一个球的重量是相等的，而这个异类球可能比其他轻或者重，现在提供一个天平，要求你通过最多三次天平测量，找出哪个球重量和其他不一样，并得出是轻还是重的结论。

仔细想一下发现做一次天平测量相当于做一次 linear measurement，将球编号为 $x_1, \ldots, x_{12}$，排成一个向量记为 x 的话，做一次测量相当于用 x 去内积一个向量 w，其中 w 的元素可能是 ± 1 或者 0。-1 表示放在天平左边，$+1$ 放在天平右边。用 m 表示正常球的质量，$M \neq m$ 表示异常球的质量，则

$$x = m\mathbf{1} + (M - m)e_k$$

其中 $\mathbf{1}$ 是一个全 1 的向量，而 e_k 是第 k 个位置为 1，其他位置为零的向量，这里假设第 k 个球是异类球，但是 k 的值目前未知。注意我们在测量的时候一定是会在左边和右边放上同样数目的球，否则测出的结果

一般情况下得不到任何有用的信息：所以 w 向量里正负 1 的数目应该是一样多的，于是 $\langle w, \mathbf{1} \rangle = 0$，也就是说：$\langle w, x \rangle = (M - m)w_k$。

题目要求我们使用三次测量，分别记为 w^1, w^2, w^3，排成矩阵为[1]

$$W = \begin{pmatrix} w^1 & w^2 & w^3 \end{pmatrix}^T$$

我们用 $W_{:j}$ 表示 W 矩阵的第 j 列，则

$$Wx = (M - m)W_{:k}$$

由于我们 w 的编码中 -1 表示放左边，$+1$ 表示放右边，所以如果左边重，我们的观察值 $\langle w, x \rangle$ 会是负数，右边重是正数，一样重是零。由于我们的天平实际上不是刻度天平，所以我们的观察值实际上是[2]

$$\operatorname{sign}(Wx) = \operatorname{sign}\left((M - m)W_{:k}\right) = \operatorname{sign}(M - m)W_{:k}$$

也就是说，我们的观察值实际上将会是矩阵的第 k 列或者该列的相反数，具体取决于异类球的质量是比正常球大还是小。所以说，如果我们能将矩阵 W 设计为任意两列以及它们的相反数都互不相等的话（当然还必须满足每一行的 ± 1 个数相等），就可以根据这个测量矩阵找出异类球的位置 k 了。具体的做法就是直接拿测量结果去和 W 的每一列比较，如果和其中第 k 列相等的话，那么异类球就是第 k 个，并且重量比其他球大；否则再用测量结果的相反数去和 W 的每一列比较，相同的那一列对应的就是异类球的标号，并且此时异类球的质量比普通球要小。

那么这样的矩阵究竟能不能构造出来呢？首先矩阵每一个元素可能的取值范围是三个，矩阵有 3 行，所以所有可能的不同的列的个数为 $3^3 = 27$ 个，除去全零（永远不测量该球）这种极端的情况，还剩 26 种，由于每一种情况和它的相反数的情况对应起来，因此我们实际上会得到 13 种，任选 12 种填满这十二列。下面是一个例子：

$$W = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 & -1 & 1 & -1 & 0 & 1 & -1 & 1 \\ 0 & 1 & 0 & -1 & -1 & 0 & -1 & 0 & 1 & 1 & 1 & -1 \\ 0 & 0 & 1 & 0 & -1 & -1 & 0 & 1 & -1 & -1 & 1 & 1 \end{pmatrix}$$

类似的算法可以推广到做 N 次测量，可以在多少个球中找出异类球来上，就不在这里再具体说了。所以这和 Compressive Sensing 有什么关系呢？当然在很多地方都是很相似的。首先 x 是一个 “稀疏” 的向量（除了一个位置之外其他的位置值都一样），然后现在通过数目远小于 x 的维度的测量来恢复出原始的 x 来。其中有一点和普通的 CS 不一样的是：这里并不是直接做原来的线性测量，而是在通过一个矩阵进行线性投影之后，测量结果的符号：$\operatorname{sign}(Wx)$，稍微忽略掉等于零的情况的话，一

[1] 题目并没有限制不能使用 adaptive 的测量方法，也就是说，根据上一次测量的结果动态地决定接下来测量那些球。但是我们这里采用 non-adaptive，也就是事先就把三次测量全部定下来的方式。

[2] 所以说我们刚才说的 “测量是线性的” 这个说法实际上是需要纠正的。

个符号 ± 1 可以用一个 bit 来表示，所以这样的特殊的设定通常称为 1-bit Compressive Sensing。

由于 sign 是一个非线性函数，所以这里的测量也是和普通 CS 不同的非线性测量。问题的设定也有些不同，因为很容易可以看到的是：原始向量 x 的长度信息永久丢失了，因为对于任意正数 $\alpha > 0$，$\text{sign}(\alpha Wx) = \text{sign}(Wx)$，所以通常情况下会尝试恢复一个单位长度的向量 x'。更常见的设定是只寻找 x 的非零元素的位置，而不去管具体的值，比如对应我们刚才的问题，就是找出 k 是多少，通常叫做 support recovery。这通常被认为是更加 achievable 的目标。

在 1-bit CS 里，除了普通 CS 里常用的随机高斯或者伯努利测量矩阵之外，由于问题设定和 Computer Science 里的 Group Testing 问题也联系更加紧密一些，所以还有许多通过巧妙地构造出来的特殊矩阵配合上专门的恢复算法（而不是简单的 ℓ_1-norm 最小化）可以来完成给定的目标。

另外，介于 1-bit CS 和普通的 CS 之间还可以有 K-bits CS，因为我们在电子计算机上处理数据的时候实际上是没有用无限精度来表示测量值的，总是在一定程度上做了 quantization，这个 quantization 的步骤（也就是用 K 个 bit 来近似真正的观察值）会产生怎样的影响，以及 K 的取值会对结果精度产生什么样的 trade-off，这也是目前 CS 里正在被研究的一个问题。