Coordinate Descent Method for k-means

最近关注到了几篇采用Coordinate descent 实现聚类的方法。然而，在谱聚类上进行加速的文章 Fast Clustering by Directly Solving Bipartite Graph Clustering Problem有些难以理解。因此，我希望回到Coordinate Descent Method for k-means， 这篇写作更清晰一些的文章来进行梳理。

Motivation

在这里，传统的k-means采用2-step的方法进行聚类；类似于EM算法，它迭代地更新聚类中心(由类似元素平均获得)和类别分配情况。
它的优化目标如下所示：

$$\min_{F\in Ind,M}\sum_{j=1}^{c}\sum_{i=1}^{n}\|\boldsymbol{x}_i-\boldsymbol{m}_j\|_2^2\boldsymbol{f}_{ij}=\min_{F\in Ind,M}\|X-MF^T\|_F^2,$$

可以看到，其类似于一个矩阵分解任务，其中M和F分别代表clustering center和index矩阵。可以这么理解： c个聚类中心存在于$\boldsymbol{m}_j$中，采用index矩阵来为每个样本点分配一个聚类中心，用ℓ_F范数处理二者的差异。Lloyd提出了一个迭代的思想来解决这个问题，也是我们最常见的k-means的思想。
其核心求解如下所示：

1. 随机确定 c 个初始点作为质心。
2. 将数据集中的每个点分配到一个簇中, 具体来讲, 就是为每个点找到距其最近的质心, 并将其分配该质心所对应的簇.
3. 每个簇的质心更新为该簇所有点的平均值.

重复上述过程直到数据集中的所有点都距离它所对应的质心最近时结束。

Pros & Cons

Pros:

k-means的好处主要是方便，通常情况下可以找到空间中聚集的模式，对于可以很好的聚类出空间中位置相近的样本；同时相比于谱聚类而言其时间开销较小。(n^2c vs $tnmc$), n是样本个数，c是类别数，m是特征维度，t是k-means的迭代次数。当簇近似高斯分布的时候，可以有不错的效果。
同时，k-means方法也可以被用于寻找锚点，此时所选取的锚点个数常常大于类别个数，从而可以辅助谱聚类或核聚类实现更高效的计算。

Cons:

k-means 的问题也很明显：

1. 首先，需要依赖于初始化，所以可能会收敛到次优解/空类别的情况；收敛速度不好保证。
2. 只能处理距离上接近的数据，对于更复杂的数据 (如存在异常点 / Two moon 数据)效果不好。
3. 需要知道类别数目c.

初始化可能是大部分聚类都存在的问题；从k-means的初始类别选取，到谱聚类或核聚类的关系度量，初始化一直都是一个问题。这篇文章关注于第一个问题: 多视角聚类对于初始化敏感，常常会有空类别/类别的问题。
希望采用Coordinate Descent的方式来逐个分配样本点，加速优化(这个如何证明)的同时可以避免空类别。

在这里，overall objective如下：

$$\max_{F\in Ind}obj(F)=\max_{k\in\{1,\ldots,c\}}\sum_{l=1}^{c}\frac{\boldsymbol{f}_l^TX^TX\boldsymbol{f}_l}{\boldsymbol{f}_l^T\boldsymbol{f}_l}.$$

如何理解? 直观感觉是k-means和谱聚类的结合。在这里在k-means的基础上加入了$\boldsymbol{f}_l^T\boldsymbol{f}_l$作为分母, 类似于Ncut的形式，采用类别中元素个数来做分母，希望模型分类较为均匀，避免分出空的类别。

然而，这个方法直观而言不方便优化。这是一个类似瑞丽熵的形式，F矩阵必须还是index matrix(每一行有一个值是1，其他是0)，更加难以处理，没法用之前所说的two-step的方式进行处理。因此，基于 coordinate descent 的思想，考虑能否每次更新一个样本，而非像Ncut一样直接更新全部样本。 (这里如果采用类似于谱聚类的方法去做，首先更新全部样本，再用k-means得到F，时间代价会很高，n^2级别。所以这里结合了k-means的优势来保证效率)。

由于每次只更新一个样本，可以对于分子$\boldsymbol{f}_l^TX^TX\boldsymbol{f}_l$和分母$\boldsymbol{f}_l^T\boldsymbol{f}_l$分别维护；容易发现: 当每个类别变化时，对于每一类而言，相比于没有分类，当它被分为某一类时，分母只有$+1/-1$的变化；而分子只有一个向量的内积的变化。如当 $f_k^{(i)}=1$(即更新第i的样本，讨论其被分于第k个类别)时，$X\boldsymbol{f}_l=X\boldsymbol{f}_l(0)+x^{(i)}$, $\boldsymbol{f}_l^T\boldsymbol{f}_l=\boldsymbol{f}_l(0)^T\boldsymbol{f}_l(0)+1$。这里用$(0)$来表示当不分配第$i$个样本时得到的结果，即对于不同的$l$而言，$\boldsymbol{f}_l$的第$i$行都为0值。受到这个启发，我们发现可以采用一种新的objective来迭代的更新每一行元素。在这里主要的变化就是引入了当第i行为0时的objective，并且采用做差取最大的方式，来反映分配权重后对于目标的增益，并且可以通过维护上式，实现加速计算。这个目标函数可以写成下面的形式，即每次对第i个样本而言：

$$\begin{aligned}&\max_{k\in\{1,\ldots,c\}}\psi(k)=\max_{k\in\{1,\ldots,c\}}\left(obj(F(k))-obj(F(0))\right)\\&=\max_{k\in\{1,\ldots,c\}}\sum_{l=1}^{c}\left(\frac{\boldsymbol{f}_l^T(k)X^TX\boldsymbol{f}_l(k)}{\boldsymbol{f}_l^T(k)\boldsymbol{f}_l(k)}-\frac{\boldsymbol{f}_l^T(0)X^TX\boldsymbol{f}_l(0)}{\boldsymbol{f}_l^T(0)\boldsymbol{f}_l(0)}\right)\end{aligned}$$

$$= \max_{k\in\{1,\ldots,c\}} \left(\frac{\boldsymbol{f}_k^T(k) X^T X \boldsymbol{f}_k(k)}{\boldsymbol{f}_k^T(k)\boldsymbol{f}_k(k)} - \frac{\boldsymbol{f}_k^T(0) X^T X \boldsymbol{f}_k(0)}{\boldsymbol{f}_k^T(0)\boldsymbol{f}_k(0)} \right).$$

在这里，因为是考虑第i个样本，所以这里提出了k个不同的可能的$F(k), k \in \{1,\ldots,c\}$. 其中$F(k)$和$F(j)$中除了第$i$行，其他的行是相同的，只有第$i$行，即第$i$个样本的类别有差别，$F(k)$表示第$k$个值是1，其他是0，代表着被分到第$k$类。此外，用$F(0)$来表示第$i$行全为0时的矩阵。所以，对于更新，由于只存在第$k$类的差别，所以只需要去动态的考虑$X\boldsymbol{f}_k(k)$和$\boldsymbol{f}_k^T(k)\boldsymbol{f}_l(k)$即可。这里可以根据这两个参数来得到这个新的objective:

$$\max_{k\in\{1,\ldots,c\}} \psi(k) = \begin{cases} \dfrac{\boldsymbol{f}_k^T X^T X \boldsymbol{f}_k}{\boldsymbol{f}_k^T \boldsymbol{f}_k} - \dfrac{\boldsymbol{f}_k^T X^T X \boldsymbol{f}_k - 2\boldsymbol{x}_i^T X f_k + \boldsymbol{x}_i^T \boldsymbol{x}_i}{\boldsymbol{f}_k^T \boldsymbol{f}_k - 1}, & k = p, \\ \dfrac{\boldsymbol{f}_k^T X^T X \boldsymbol{f}_k + 2\boldsymbol{x}_i^T X f_k + \boldsymbol{x}_i^T \boldsymbol{x}_i}{\boldsymbol{f}_k^T \boldsymbol{f}_k - 1} - \dfrac{\boldsymbol{f}_k^T X^T X \boldsymbol{f}_k}{\boldsymbol{f}_k^T \boldsymbol{f}_k}, & k \neq p. \end{cases}$$

比较容易理解，这个式子就是直接对于两种情况进行讨论得到的。得到了不同的 $\psi(k)$ 值后只需要对于不同类别取最大值就好了。这样依次迭代地对于所有数据进行，直到收敛，就可以得到最后的聚类结果。由下面式子可以得出，复杂度为$O(ndct + nd + cd)$, 其中t是迭代次数，d是表征维度。直观而言，其也是达到了k-means聚类的所能达到的最优复杂度了，因为无论如何都是要尝试把数据放入每一类的，就需要ndc的复杂度了。想要继续加速，可能需要一些anchor相关的方法，或者多次聚类的方式再进行优化。同时，该方法无须初始化类别，解决了上述问题；同时因为其只需要依次给每个样本分配就好了，这样也可以加速计算。
所有这篇文章带来的方法还是非常有意义的，后续也被扩展到谱聚类中，在这里由于十分相似，就不详述了。

Algorithm 4. A Fast Version of CD to Solve Problem (10)

Input data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$, cluster number c.
Initialize c cluster centers by an initial strategy and get initial $\mathbf{F} \in \mathbb{R}^{n\times c}$.
Compute and store $\mathbf{X}f_k$, $f_k^{\mathrm{T}} f_k$, $f_k^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}f_k (k = 1, 2, \ldots, c)$, and $x_i^{\mathrm{T}} x_i (i = 1, 2, \ldots, n)$.
repeat
  for $i = 1$ to n **do**
    Calculate $\psi(k)(k = 1, 2, \ldots, c)$ by Eq. (15);
    Update the ith row of $\mathbf{F}$ by Eq. (16);
    if $p \neq q$ **then**
      Update $\mathbf{X}f_k$ and $f_k^{\mathrm{T}} f_k (k = p, q)$ by Eq. (17);
      Update $f_k^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}f_k (k = p, q)$ by Eq. (18).
    end if
  end for
until convergence
Output indicator matrix $\mathbf{F} \in \mathbb{R}^{n\times c}$.

Some insights

1. 该方法可以准确的学出来一个index; 但是在多视角或带噪数据中，通常仅仅用index矩阵来表示其聚类结果可能是不准确的。过去k-means会采用fuzzy的方式来解决这一问题；在这里，作者通过argmax来直接得到硬边界；这可能很难处理带噪的样本。所以或许能否将fuzzy引入是一个要点。这里我后面再去推导一下。
2. 如何评价它和k-means的好坏呢？似乎过去的文章主要在于设计新的算法，而很少有人关心如何保证算法是否能收敛/是否优于另一个算法/在什么条件下优于另一个算法。
3. 这里采用了Coordinate Descent的思想进行实验；感觉这有些类似于随机梯度下降(SGD也是对于每个样本/batch进行优化，能获得更好的局部解)；如果结合第二点可以证明如果随机的CD可以收敛到更好的界，并给出bound，可能是一个不错的文章!