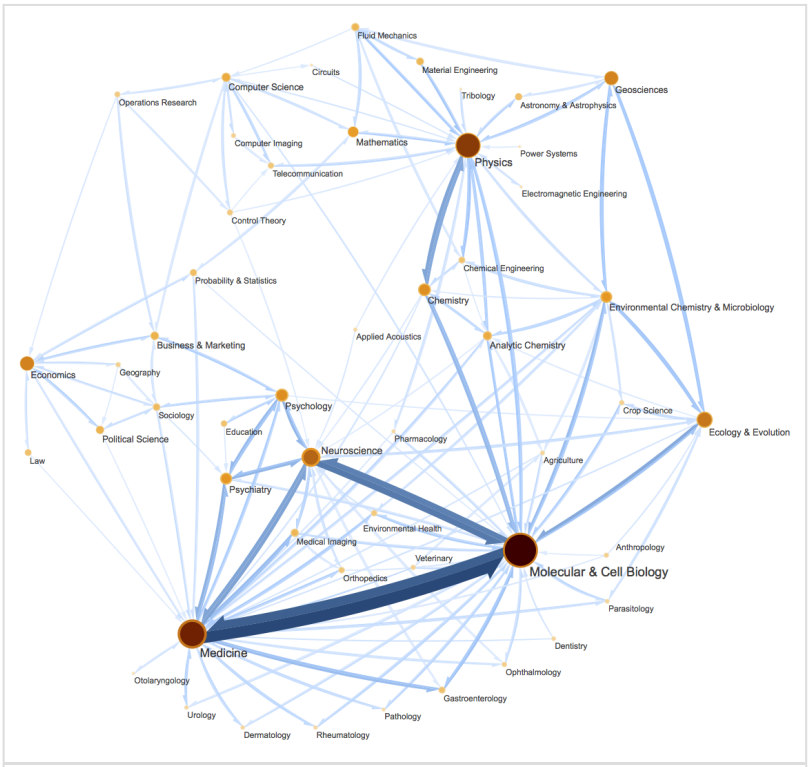
19 最小熵原理（五）：“层层递进”之社区发现与聚类

Oct By 苏剑林 | 2019-10-19 | 13911位读者

让我们不厌其烦地回顾一下：最小熵原理是一个无监督学习的原理，“熵”就是学习成本，而降低学习成本是我们的不懈追求，所以通过“最小化学习成本”就能够无监督地学习出很多符合我们认知的结果，这就是最小熵原理的基本理念。

这篇文章里，我们会介绍一种相当漂亮的聚类算法，它同样也体现了最小熵原理，或者说它可以通过最小熵原理导出来，名为InfoMap，或者MapEquation。事实上InfoMap已经是2007年的成果了，最早的论文是《Maps of random walks on complex networks reveal community structure》，虽然看起来很旧，但我认为它仍是当前最漂亮的聚类算法，因为它不仅告诉了我们“怎么聚类”，更重要的是给了我们一个“为什么要聚类”的优雅的信息论解释，并从这个解释中直接导出了整个聚类过程。



一个复杂有向图网络示意图。图片来自InfoMap最早的论文《Maps of random walks on complex networks reveal community structure》

当然，它的定位并不仅仅局限在聚类上，更准确地说，它是一种图网络上的“社区发现”算法。所谓社区发现（Community Detection），大概意思是给定一个有向/无向图网络，然后找出这个网络上的“抱团”情况，至于详细含义，大家可以自行搜索一下。简单来说，它跟聚类相似，但是比聚类的含义更丰富。（还可以参考《什么是社区发现?》）

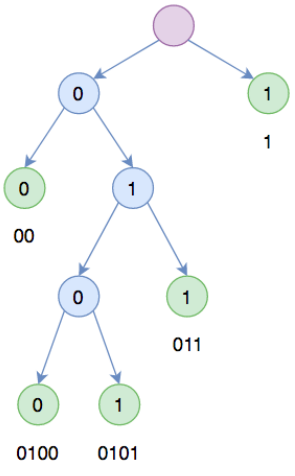
熵与编码

在前面几篇文章中，我们一直用信息熵的概念来描述相关概念，而从这篇文章开始，我们引入信息熵的等价概念——平均编码长度，引入它有助于我们更精确地理解和构建最小熵的目标。

二叉树编码

所谓编码，就是只用有限个标记的组合来表示原始信息，最典型的就是二进制编码，即只用0和1两个数字。编码与原始对象通常是一一对应的，比如“科”可能对应着11，“学”对应着1001，那么“科学”就对应着(11,1001)。

注意这里我们考虑的是静态编码，即每个被编码对象跟编码结果是一一对应的，同时这里考虑的是无分隔符编码，即不需要额外的分隔符就可以解码出单个被编码对象。这只须要求每个编码不能是其余某个编码的前缀（这样我们每次只需要不断读取编码，直至识别出一个编码对象，然后再开始新的读取）。这就意味着所有编码能够构建成一棵二叉树，使得每个编码对应着这棵树的某个叶子，如图所示。



每个编码都不是其余某个编码的前缀，所以这些编码可以构成一棵二叉树

在此基础上，我们可以得到一个很有趣也很有意义的结论，就是假设n个不同的字符，它们的编码长度分别为$l_1, l_2, \cdots, l_n$，那么我们有

$$\sum_{i=1}^{n} 2^{-l_i} \leq 1 \tag{1}$$

这其实就是这种二叉树表示的直接推论了，读者可以自行尝试去证明它。

最短编码长度

现在想象一个“速记”的场景，我们需要快速地记录下我们所听到的文字，记录的方式正是每个字对应一个二进制编码（暂时别去考虑人怎么记得住字与编码间的对应关系），那为了记得更快，我们显然是希望经常出现的字用短的编码，比如“的”通常出现的很频繁，如果用11000010001来表示“的”，那么每次听到“的”我们都需要写这么一长串的数字，会拖慢记录速度，反而如果用短的编码比如10来表示“的”，那相对而言记录速度就能有明显提升了。

假设一共有n个不同的字符，每个字符的出现概率分别是$P = (p_1, p_2, \cdots, p_n)$，那么我们可能会感兴趣两个问题：**第一是如何找到最优的编码方案，使得总的平均编码长度最短；第二是理论上这个平均编码长度最短是多少？**

对于第一个问题，答案是Huffman编码，没错，就是Word2Vec里边的Huffman Softmax的那个Huffman，但这不是本文的重点，有兴趣的读者自行找资料阅读就好。而第二个问题的答案，是信息论里边的一个基本结果，它正是**信息熵**

$$H(P) = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{2}$$

也就是说，信息熵正好是理论上的最短平均编码长度。注意这是一个非常“强大”的结果，它告诉我们不管你用什么编码（不管是有分隔符还是没分隔符，不管是静态的还是动态的），其平均编码长度都不可能低于(2)。

在这里我们就前面说的无分隔符编码场景，给出(2)的一个简单的证明。依然设n个字符的编码长度分别是$l_1, l_2, \cdots, l_n$，那么我们可以计算平均编码长度：

$$\sum_{i=1}^{n} p_i l_i = -\sum_{i=1}^{n} p_i \log_2 2^{-l_i} \tag{3}$$

因为我们有不等式(1)，所以定义：

$$\hat{p} = 1 - \sum_{i=1}^{n} 2^{-l_i} \geq 0 \tag{4}$$

上式可以写成：

$$-0 \times \log \hat{p} - \sum_{i=1}^{n} p_i \log_2 2^{-l_i} \tag{5}$$

这就是概率分布$P = (0, p_1, p_2, \cdots, p_n)$和$Q = (\hat{p}, 2^{-l_1}, 2^{-l_2}, \cdots, 2^{-l_n})$的交叉熵，而交叉熵在$P = Q$时取得最小值（这可以通过$KL(P\|Q) \geq 0$得到），所以最终有

$$\sum_{i=1}^{n} p_i l_i \geq -\sum_{i=1}^{n} p_i \log_2 p_i \tag{6}$$

其实稍微接触过信息论的读者，都应该知道这个结论，它告诉我们编码的最短平均长度就是信息熵，这其实也是无损压缩的能力极限，我们通过寻找更佳的方案去逼近这个极限，这便是最小熵。

最后，由于不同的对数底只会导致结果差一个常数比例，所以通常情况下我们不特别声明是哪个底，有些情况下则干脆默认地使用自然对数底，因为自然对数有时能得到更简单的理论形式。

InfoMap

回到InfoMap，它跟前面说的压缩编码有直接联系。事实上，在中大读研一的时候导师就已经给笔者介绍过InfoMap了，但是很遗憾，直到前几天（2019年10月15日）我才算是理解了InfoMap的来龙去脉。那么久都没弄懂的原因，一方面是当时对信息熵和编码相关理论都不熟悉，以至于看到文章的概念就觉得迷迷糊糊；另一方面，我觉得原论文的作者们可能并不清楚非信息论相关的读者理解这个算法的瓶颈在哪，以至于没有把关键地方表达清楚。

于是我决定把我自己的理解写下来，希望能让给多的读者更好地理解InfoMap这个算法，毕竟它真的很美～～

InfoMap的论文清单：https://www.mapequation.org/publications.html

（为了一个算法还特意办了一个网站，并且算法活跃至今，可见作者本身对这个算法的热爱，以及这个算法的漂亮和有效。）

分类记忆

假如我们有这么一个任务，要求我们在短时间内把下面的序列背下来：

雪梨	葡萄	香蕉	上海	广州	北京	杭州	深圳	123	654	798	963

待记忆序列

序列不长，背下来不难，为了快速地形成记忆，大家的记忆思路可能是这样的：

> 1、前面3个是水果；中间5个是城市；后面4个是阿拉伯数字；
>
> 2、前面3个水果分别是雪梨、葡萄、香蕉；
>
> 3、中间5个城市分别是广州、上海、北京、杭州、深圳；
>
> 4、后面4个数字分别是123、654、798、963。

也就是说，大家基本上都会想着按类分组记忆的思路，这样记忆效率和效果都会更好些，而最小熵系列告诉我们，“提效”、“省力”的数学描述就是熵的降低，所以一个好的分类方案，应该是满足最小熵原理的，它能够使得系统的熵降低。这便是InfoMap本质的优化目标，通过最小化熵来寻求最优的聚类方案。

层次编码：概念

前面说了，信息熵就等价于最短编码长度，所以最小熵事实上就是在寻找更好的压缩算法。既然刚才说分组记忆效率更高，那必然对应着某种编码方法使得我们能更有效地压缩信息，这种编码方法就是层次编码。

所谓层次编码，就是我们不再用单一的编码来表示一个对象，而是用两个（也可以更多，本文主要分析两个）编码的组合来表示一个对象，其中第一个编码代表对象的类别，第二个编码则代表它在类内的编号。假如同一个类的对象经常“扎堆”出现，那么层次编码就能起到压缩的作用。

究竟怎样的层次编码能压缩呢？很遗憾，我看到的所有InfoMap的资料，包括原作者的论文，都没有突出这个分层编码的方法。我认为这个编码过程是要突出的，这样对非信息论相关专业的读者会更友好。事实上，它的编码过程如下：

水果	雪梨	葡萄	香蕉	End	城市	上海	广州	北京	杭州	深圳	End	数字	123	654	798	963	End
000	001	010	011	000	001	001	010	011	100	101	000	011	001	010	011	100	000

层次编码方式。在同一个类别的词语前插入一个类别标记，以及在类结束处插入一个终止标记

要注意，编码要保证无损，换言之要有相应的方法解码为原始序列，为此层次编码在原始序列的基础上插入了类别标记和终止标记，其中类别标记用单独一套编码，类别内的对象以及终止标记用另一套编码，由于有了类别区分，因此不同类的对象可以用同一个编码，比如上图中“雪梨”和“上海”都可以用001编码，这样就可以使得整体的平均编码长度变小了。解码的时候，只需要读到第一个类别编码，就开始使用该类别的类内编码来识别原对象，直到出现结束符就开始新的类别编码读取，然后重复这个过程。

层次编码：计算

好了，既然确定了这种编码方式确实是无损的，接下来就需要计算了，因为“缩短平均编码长度”的结论不能单靠直观感知，还必须有定量的计算结果。

假设已经有了特定的层次编码方案，那么我们就可以计算这种编码方案的平均编码长度了，定义下述记号

记号	含义
p_α	对象α的出现概率
$q_{i\curvearrowright}$	类别i的出现概率

(7)

根据上述编码约定，每个类别序列的出现，必然以该类别的终止标记结束，所以$q_{i\curvearrowright}$也是类别i的终止标记的出现概率。要强调的是，这里的概率都是指全局归一化的概率，也就是

$$\underbrace{\sum_i q_{i\curvearrowright}}_{\text{类别的概率总和}} + \underbrace{\sum_\alpha p_\alpha}_{\text{编码对象的概率总和}} + \underbrace{\sum_i q_{i\curvearrowright}}_{\text{终止标记的概率总和}} = 1 \tag{8}$$

由于类和类内对象使用两套不同的编码，所以可以分别计算两者的最短平均编码长度。其中，类的最短平均编码长度是

$$H(\mathcal{Q}) = -\sum_i \frac{q_{i\curvearrowright}}{q_{\curvearrowright}} \log \frac{q_{i\curvearrowright}}{q_{\curvearrowright}} \tag{9}$$

这里$q_{\curvearrowright} = \sum_i q_{i\curvearrowright}$。这是因为类的编码是独立一套的，将类的概率在类间归一化后，代入熵的公式就得到类的理论上的最短平均编码长度了。

类似地，每个类i的类内对象的最短平均编码长度是

$$H(\mathcal{P}^i) = -\frac{q_{i\curvearrowright}}{p_{i\circlearrowright}} \log \frac{q_{i\curvearrowright}}{p_{i\circlearrowright}} - \sum_{\alpha\in i} \frac{p_\alpha}{p_{i\circlearrowright}} \log \frac{p_\alpha}{p_{i\circlearrowright}} \tag{10}$$

这里$p_{i\circlearrowright} = q_{i\curvearrowright} + \sum_{\alpha\in i} p_\alpha$。对了，要提醒读者的是**认真区分好记号**（$p$还是$q$？$\curvearrowright$还是$\circlearrowright$？），不要看错了，一旦看错将会大大增加你的理解难度。上式的含义也很清晰，每个类的类内对象都用独自的一套编码，所以连同终止标记概率在类内归一化后，代入熵的公式。

最后，将两者加权平均，就得到总的最短平均编码长度了：

$$L(M) = q_{\curvearrowright} H(\mathcal{Q}) + \sum_i p_{i\circlearrowright} H(\mathcal{P}^i) \tag{11}$$

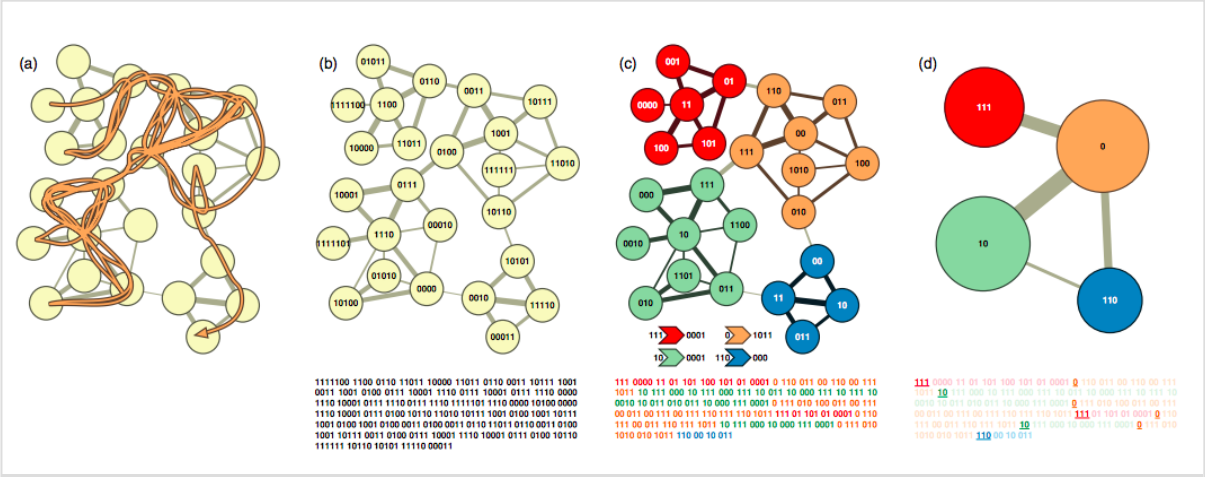
现在，我们有了一个定量的指标(11)，可以比较两种不同的层次编码方案的优劣。反过来，有了一个对象序列后，我们可以去搜寻一个层次编码方案，使得(11)越小越好，从而就找到了这些对象的一个聚类方案。注意这样的聚类算法几乎没有任何超参数——只需要给定对象序列，我们就可以得到该序列的层次编码，目标是(11)最小化，这不需要事先指定聚类数目等参数（管你聚多少类，总之只要(11)更小就行了）。

随机游走

至此，我们得到了一个优化目标(11)，在给定一个序列的情况下，我们可以通过最小化这个目标来为这个序列的对象聚类。但我们仍然还没有将它跟经典的聚类任务对应起来，因为经典的聚类问题并没有这种序列，一般只是给出了任意两个样本之间的相似度（或者给出样本本身的向量，通过这个向量也可以去算两个样本之间的某种相似度）。

InfoMap做得更细致一些，它将我们要聚类的样本集抽象为一个有向图，每个样本是图上的一个点，而图上的任意两个点(α, β)都有一条$\beta \to \alpha$的边，边的权重为转移概率$p_{\beta\to\alpha}$。对于普通的图，图上的边一般就是代表着两个点之间的相似度，然后我们可以通过将边的权重归一化来赋予它转移概率的含义。

有了这个设定之后，一个很经典的想法——“随机游走”——就出来了：从某个点j开始，依概率$p(i|j)$跳转下一个点i，然后从i点出发，再依转移概率跳转到下一个点，重复这个过程。这样我们就可以得到一个很长很长的序列。有了序列之后，就能以(11)为目标来聚类了。



InfoMap编码与聚类过程。(a)随机游走; (b)根据随机游走的概率直接构建huffman编码; (c)层次编码; (d)层次编码中的类别编码。最下方显示了对应的编码序列，可以看到层次编码的编码序列更短

这就是InfoMap的聚类过程了——**通过构造转移概率，在图上进行随机游走来生成序列，在通过对序列做层次编码，最小化目标(11)，从而完成聚类。**

顺便提一下，当初DeepWalk（2014年）提出来在图上进行随机游走生成序列然后套用Word2Vec的做法，不少人都觉得很新奇，是个突破，但现在看来在图上做随机游走的思路由来已久了。

InfoMap

现在，我们将上述过程做得更细致化、数学化一些，毕竟数学公式是把原理描述清楚的唯一方法。

首先，我们不需要真的图上进行随机游走模拟，因为事实上我们不需要生成的序列，我们只需要知道随机游走生成的序列里边每个对象的概率，为此，我们只需要去解方程

$$\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix} = \begin{pmatrix} p_{1\to 1} & p_{2\to 1} & \cdots & p_{n\to 1} \\ p_{1\to 2} & p_{2\to 2} & \cdots & p_{n\to 2} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1\to n} & p_{2\to n} & \cdots & p_{n\to n} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix} \tag{12}$$

或者写成$p_\beta = \sum\limits_\alpha p_\alpha p_{\alpha\to\beta}$。这也好理解，也就是假设最终的平稳分布是$(p_1, p_2, \cdots, p_n)$，那么再随机游走一步，它还是原来的分布。

但是，单纯这样的随机游走，结果可能会依赖于迭代的初始值，说白了，就是方程(12)的解可能不唯一。这并不难想象，假如图上有一些孤立区域，那如果游走进这些孤立区域，那就永远出不来了，导致区域以外的点的概率全都为0了。结果跟初始值有关，这是不合理的，聚类结果应该只跟图本身有关。为了解决这个问题，作

者引入了“穿越概率”τ：以$1-\tau$的概率按照$p_{\beta\to\alpha}$的转移概率来随机游走，以τ的概率随机选择图上任意一个点跳转。这样的话，p_α的方程是

$$p_\beta = (1-\tau)\sum_\alpha p_\alpha p_{\alpha\to\beta} + \tau\sum_\alpha \frac{p_\alpha}{n} \tag{13}$$

其实就相当于转移概率$p_{\alpha\to\beta}$换成了$(1-\tau)p_{\alpha\to\beta}+\tau/n$。

有了穿越概率，模型一般就不会陷入某个局部解了，从而导致了解的唯一性。τ是一个额外的超参，作者取了$\tau=0.15$，而为了使得结果对τ更鲁棒，作者还使用了其他一些技巧，具体细节请去看作者的《The map equation》和《Community detection and visualization of networks with the map equation framework》。

现在，p_α有了，距离目标(11)，我们还差$q_{i\curvearrowright}$，$q_{i\curvearrowright}$是类别的概率，也是终止标记的概率。什么时候会出现终止标记？出现终止标记意味着后面的类是已经不是i类了，所以终止标记的概率实际上就是“从一个类别跳转到另一个类别的概率”，换言之就是随机游走过程中“离开i类”这件事情发生的概率，它等于

$$q_{i\curvearrowright} = \sum_{\alpha\in i}\sum_{\beta\notin i} p_\alpha p_{\alpha\to\beta} \tag{14}$$

如果带有穿越概率的话，那要将$p_{\alpha\to\beta}$换成$(1-\tau)p_{\alpha\to\beta}+\tau/n$，即

$$\begin{aligned} q_{i\curvearrowright} =& \sum_{\alpha\in i}\sum_{\beta\notin i} p_\alpha\left[(1-\tau)p_{\alpha\to\beta}+\frac{\tau}{n}\right] \\ =& \tau\frac{n-n_i}{n}\sum_{\alpha\in i} p_\alpha + (1-\tau)\sum_{\alpha\in i}\sum_{\beta\notin i} p_\alpha p_{\alpha\to\beta} \end{aligned} \tag{15}$$

其中n_i是类i的类内对象数目。

现在p_α和$q_{i\curvearrowright}$的形式都出来了，理论上我们需要将p_α和$q_{i\curvearrowright}$按照式(8)归一化，但是(15)的形式显示$q_{i\curvearrowright}$也是正比于p_α的，而优化目标(11)只看相对概率，所以归不归一化都行。

好，现在万事具备，InfoMap整个流程就出来了（第3步的搜索策略在实验部分再介绍）：

1、定义好样本间的转移概率$p_{\alpha\to\beta}$；

2、数值求解(13)，得到p_a；

3、搜索聚类方案，使得(11)尽可能小，其中每种聚类方案中的$q_{i\curvearrowright}$按照(15)算。

推广思路

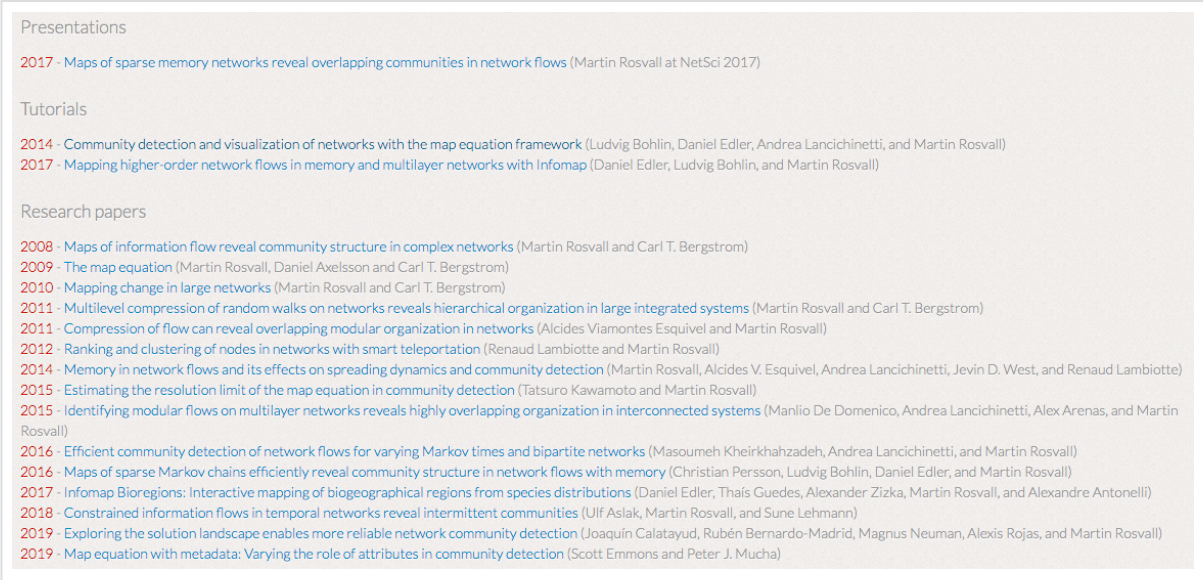
InfoMap的美妙之处，不仅在于它漂亮的信息论解释、没有过多的超参等，还在于它易于推广性——至少思路上是很容易想到的。

比如，前面都是用两层的层次编码，我们可以构造多层的层次编码，也就是给类再聚类，而事实上将优化目标(11)从双层推广到多层只是一件繁琐但没有技术难度的事情，因为只需要模仿双层编码的方式，构思对应的多

层编码的方案（引入类中类的类标记和终止标记等），具体细节可以参考《Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems》、《Community detection and visualization of networks with the map equation framework》，本文不再详细介绍了。

再比如，还可以推广到重叠社区挖掘。所谓重叠社区挖掘，也就是单个对象可能同时属于多个不同的类别。《什么是社区发现?》一文提供一个关于小区大妈组队跳广场舞的很生动的例子：一般来说，每个大妈参加活动时，往往会优先选择自己所在小区的广场舞队伍，但是，有些大妈精力比较旺盛，觉得只参加自己小区的队伍还不过瘾，或者她是社交型人士，希望通过跳广场舞认识更多的朋友，那么，她可能会同时参加周边小区的多个广场舞队伍，换言之，她同时就属于多个社区（类）了。另外还可以从NLP的词聚类角度来看，重叠社区挖掘就意味着多义词的挖掘了。对于InfoMap来说，挖掘重叠社区依然是最小化(11)，将同一个对象赋予到多个类中，对该对象本身的编码会引入一定的冗余，但是有可能减少了类标记和终止标记的使用，从而降低了整体的平均编码长度。论文则可以参考《Compression of Flow Can Reveal Overlapping-Module Organization in Networks》。

这些推广特性都是大多数其他聚类算法和社区发现算法所不具备的，这进一步体现了InfoMap的强大与漂亮。此外，官网的publications页面上还有一堆InfoMap的推广和应用论文，都可以参考：



Presentations

2017 - Maps of sparse memory networks reveal overlapping communities in network flows (Martin Rosvall at NetSci 2017)

Tutorials

2014 - Community detection and visualization of networks with the map equation framework (Ludvig Bohlin, Daniel Edler, Andrea Lancichinetti, and Martin Rosvall)
2017 - Mapping higher-order network flows in memory and multilayer networks with Infomap (Daniel Edler, Ludvig Bohlin, and Martin Rosvall)

Research papers

2008 - Maps of information flow reveal community structure in complex networks (Martin Rosvall and Carl T. Bergstrom)
2009 - The map equation (Martin Rosvall, Daniel Axelsson and Carl T. Bergstrom)
2010 - Mapping change in large networks (Martin Rosvall and Carl T. Bergstrom)
2011 - Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems (Martin Rosvall and Carl T. Bergstrom)
2011 - Compression of flow can reveal overlapping modular organization in networks (Alcides Viamontes Esquivel and Martin Rosvall)
2012 - Ranking and clustering of nodes in networks with smart teleportation (Renaud Lambiotte and Martin Rosvall)
2014 - Memory in network flows and its effects on spreading dynamics and community detection (Martin Rosvall, Alcides V. Esquivel, Andrea Lancichinetti, Jevin D. West, and Renaud Lambiotte)
2015 - Estimating the resolution limit of the map equation in community detection (Tatsuro Kawamoto and Martin Rosvall)
2015 - Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems (Manlio De Domenico, Andrea Lancichinetti, Alex Arenas, and Martin Rosvall)
2016 - Efficient community detection of network flows for varying Markov times and bipartite networks (Masoumeh Kheirkhahzadeh, Andrea Lancichinetti, and Martin Rosvall)
2016 - Maps of sparse Markov chains efficiently reveal community structure in network flows with memory (Christian Persson, Ludvig Bohlin, Daniel Edler, and Martin Rosvall)
2017 - Infomap Bioregions: Interactive mapping of biogeographical regions from species distributions (Daniel Edler, Thaís Guedes, Alexander Zizka, Martin Rosvall, and Alexandre Antonelli)
2018 - Constrained information flows in temporal networks reveal intermittent communities (Ulf Aslak, Martin Rosvall, and Sune Lehmann)
2019 - Exploring the solution landscape enables more reliable network community detection (Joaquín Calatayud, Rubén Bernardo-Madrid, Magnus Neuman, Alexis Rojas, and Martin Rosvall)
2019 - Map equation with metadata: Varying the role of attributes in community detection (Scott Emmons and Peter J. Mucha)

InfoMap官方提供的论文列表

实验

好了，说了那么久理论，也该动动手做做实验了。本节会先简单介绍一下InfoMap具体是怎么搜索这种分类策略的，然后给出一个词聚类的例子，初步体验一下InfoMap的魅力。

求解算法

InfoMap最早的求解算法是贪心搜索加模拟退火。贪心搜索中，最开始将每个点都视为不同的类，然后将使得(11)下降幅度最大的两个类合并为一个类，并且重复这个过程，直到不能下降为止；模拟退火是在探索搜索的结果基础上对聚类方案进行进一步的微调。（来自《Maps of random walks on complex networks reveal community structure》）

而后在《The map equation》一文中，作者改进了贪心搜索，并移除了模拟退火，使得它达到了速度和效果的平衡。改进的算法大概是（具体细节还是得看原论文）：

1、每个点都初始化一个独立的类；

2、按照随机的顺序遍历每个点，将每个点都归到让(11)下降幅度最大的那个相邻的类；

3、重复步骤2（每次都用不同的随机顺序），直到(11)不再下降；

4、通过一些新的策略来精调前面三步得到的聚类结果。

不管是早期的求解算法还是后来的改进算法，InfoMap的求解都称得上非常快，它里边给出了两组参考结果：1、单纯用贪心搜索可以在260万个节点、2900万条边的图网络上成功完成社区发现；2、改进的算法在1万个节点、10万条边的图网络上的社区发现可以在5秒内完成（考虑到当时的计算机并没有现在的快，所以现在跑的话就不用五秒了。）。

安装说明

作者们用C++实现了InfoMap，并且提供了Python、R等第三方语言的接口，同时支持Linux、Mac OS X和Windows系统~

官网：https://www.mapequation.org/code.html
Github：https://github.com/mapequation/infomap

这里自然是只关心Python接口了。经过一番折腾，确定了下面几个情况：

1、InfoMap目前有两个版本，其中0.x的是稳定版，另外有个1.0处于beta阶段；

2、直接pip install infomap安装的就是beta的1.0版本，但是功能不全，并且只支持Python 3.x；

3、全功能的还是0.x版本，支持Python 2.7，但我不知道为什么0.x的最新版编译成功后import会出错。

因为我之前就安装过InfoMap并且用得很舒适，但弄到新版后各种问题（不知道作者是怎么改的），所以我建议还是从Github上拉它的0.x的旧版本手动编译安装吧，下面是参考的编译安装代码：

```
wget -c https://github.com/mapequation/infomap/archive/6ab17f8b18a6fdf34b2a53454f79a3b9
unzip 6ab17f8b18a6fdf34b2a53454f79a3b976a49201.zip
cd infomap-6ab17f8b18a6fdf34b2a53454f79a3b976a49201
cd examples/python
make

# 编译完之后，当前目录下就会有一个infomap文件夹，就是编译好的模块；
# 为了方便调用，可以复制到python的模块文件夹（每台电脑的路径可能不一样）中
python example-simple.py
cp infomap /home/you/you/python/lib/python2.7/site-packages -rf
```

examples/python则给出了一些简单的demo，可以阅读参考。

词聚类

下面跟Word2Vec配合，跑一个词聚类的例子，代码位于：
https://github.com/bojone/infomap/blob/master/word_cluster.py

聚类结果（部分）：

[u'妹妹', u'姐姐', u'哥哥', u'弟弟', u'爸爸', u'儿子', u'母亲', u'女儿', u'父亲', u'妻子', u'爷爷', u'老婆', u'丈夫', u'男友', u'女友', u'爱人', u'妈妈', u'长子', u'小时候', u'父母', u'祖父', u'情人', u' 亲人', u'弟', u'家人', u'夫妇', u'妻', u'兄', u'妹', u'家里', u'姐妹', u'父', u'嫁给', u'从小', u'子女', u'嫁', u'夫', u'家中', u'娶', u'姐', u'叔', u'长大', u'夫人', u'父子', u'在家', u'娘', u'兄弟', u' 家属', u'奴', u'母', u'子', u'哥', u'儿', u'儿女', u'小姐']

[u'加强', u'建立健全', u'推进', u'拟定', u'落实', u'提高', u'规章制度', u'会同', u'搞好', u'切实', u'负责', u'促进', u'贯彻落实', u'制订', u'增强', u'抓好', u'深化', u'组织协调', u'拟订', u'加快', u'加大', u'实施', u'着力', u'有利于', u'制定', u'推动', u'进一步', u'贯彻', u'督促', u'改善', u'大力', u'贯彻执行', u'充分发挥', u'牵头', u'政策措施', u'年度计划', u'强化', u'提升', u'增进', u'中长期', u'健全', u'优化', u'方针', u'做好', u'协调', u'统筹', u'承办', u'协助', u'全力', u'积极', u'责任制', u'党和国家', u'步伐', u'力度', u'承担', u'开展', u'巩固', u'编制', u'有利', u'改进', u'加深', u'不利', u'各项', u'加速', u'围绕', u'组织', u'衔接', u'认真']

[u'股权', u'证券', u'融资', u'股票', u'金融机构', u'上市公司', u'信贷', u'债券', u'商业银行', u'贷款', u'金融', u'期货', u'信托', u'存款', u'债权', u'担保', u'股份', u'银行', u'抵押', u'董事长', u'副总经 理', u'外汇', u'总经理', u'利率', u'董事', u'余额', u'总监', u'保险公司', u'并购', u'股东', u'利息', u'CEO', u'国有企业', u'债务', u'资产', u'分行', u'经理', u'负债', u'股市', u'信用', u'总额', u'总裁', u'国有资产', u'股价', u'投资者', u'资本', u'汇率', u'数额', u'金额', u'投资', u'首席', u'账户', u'国有', u'货币', u'邮政', u'董事会', u'创始人', u'改制', u'支行', u'主管', u'出资', u'破产', u'重组', u'财产', u'披露', u'固定资产', u'收购', u'上涨', u'财富', u'商业', u'现金']

[u'教师', u'教学', u'教研', u'素质教育', u'课堂教学', u'教学质量', u'教学改革', u'该校', u'学生', u'基础教育', u'我校', u'高职', u'德育', u'教职工', u'职业教育', u'高等院校', u'办学', u'教师队伍', u'院校', u'师资', u'在校生', u'考生', u'教育', u'高等学校', u'同学们', u'专任教师', u'毕业生', u'班级', u'班主任', u'报考', u'人才培养', u'高等教育', u'在校', u'孩子们', u'高校', u'老师', u'学员', u'招生', u'全 校', u'育人', u'学子', u'大学生', u'教学班', u'校长', u'入学', u'家长', u'在校学生', u'录取', u'师生', u'青少年', u'同学', u'全日制', u'心理健康', u'招收', u'课堂', u'报名', u'青年', u'校友', u'先生', u' 义务教育', u'校园', u'校', u'成绩', u'女士']

[u'慢性', u'疾病', u'糖尿病', u'急性', u'病变', u'肿瘤', u'高血压', u'治疗', u'炎症', u'并发症', u'心脏病', u'癌症', u'水肿', u'患者', u'病理', u'病因', u'腹泻', u'呕吐', u'咳嗽', u'症状', u'便秘', u'临床表现', u'出血', u'病人', u'症', u'贫血', u'手术', u'头痛', u'病例', u'疗效', u'康复', u'发病', u'综合征', u'疗法', u'诊断', u'发作', u'切除', u'病情', u'功效', u'损伤', u'病', u'传染病', u'鉴别', u'感染', u'患', u'病毒', u'瘤', u'术', u'囊', u'畸形', u'部位', u'发热', u'伴有', u'中毒']

[u'魔王', u'恶魔', u'妖', u'魔', u'冥', u'邪恶', u'吸血鬼', u'怪物', u'死神', u'鬼', u'幽灵', u'猎人', u'幻', u'魔兽', u'仙', u'黑暗', u'诅咒', u'封印', u'僵尸', u'毁灭', u'舰', u'舰队', u'玄', u'咒', u' 精灵', u'幽', u'兽', u'艘', u'魔鬼', u'凶', u'骑士', u'BOSS', u'地狱', u'复活', u'国王', u'公主', u'女王', u'神仙', u'仙人', u'诀', u'召唤', u'尸', u'逍遥', u'魔法', u'怪', u'法术', u'神', u'王子', u'女神', u'魔力', u'灵', u'勇士', u'魂', u'邪', u'天使', u'怪兽', u'化身', u'尸体', u'武士', u'海盗', u'恶', u'戒', u'预言', u'死者', u'风流', u'光明', u'副本', u'变身', u'丹', u'杀手', u'正义', u'武功', u'荒']

显然看起来聚类效果是很不错的。

总结

写了好几天，终于把这篇博客写完了，被我搁置了2年多的算法，总算是把它拿起来并且基本理解清楚了。

总的来说，InfoMap就是建立在转移概率基础上的一种聚类/社区发现算法，有清晰的信息论解释（最小熵解释），并且几乎没有任何超参（或者说超参就是转移概率的构建），目前不少领域都开始关注到它，试图用它来从数据中挖掘出有一定联系的模块（哪个领域没有节点和图网络的结构呢？）。当初我导师就是建议我用它来分析基因数据的，可惜我最终还是没有完成这个目标～不管怎样，就因为它如此的优雅美妙，我觉得都应该去好好了解它。

最后，顺便说一下，这已经是最小熵原理的第五篇文章了，它究竟还能走多远？让我们拭目以待～