Deciding the Number of Clusterings

http://freemind.pluskid.org/machine-learning/deciding-the-number-of-clusterings

posted on Free Mind on November 18, 2012
generated with pandoc on December 3, 2015
category: Machine Learning

tags: Inference, Unsupervised Learning, Probabilistic Graphical Model

如何确定聚类的类别个数这个问题经常有人问我，也是一直以来让我自己也比较困惑的问题。不过说到底其实也没什么困惑的，因为这个问题本身就是一个比较 ill posed 的问题呀：给一堆离散的点，要你给出它们属于几个 cluster，这个基本上是没有唯一解或者说是没有合适的标准来衡量的。比如如果简单地用每个类别里的点到类中心的距离之和来衡量的话，一下子就会进入到 “所有的点都独立成一类” 这样的尴尬境界中。

但是 ill posed 也并不是一个很好的理由，因为我们其实大部分时候都是在处理 ill posed 的问题嘛，比如 Computer Vision 整个一个领域基本上就没有啥问题是 well posed 的..... `=.=bb`，比如下面盗用一下 Bill Freeman 的 Slides 中的一张讲 deblur 的问题的图：

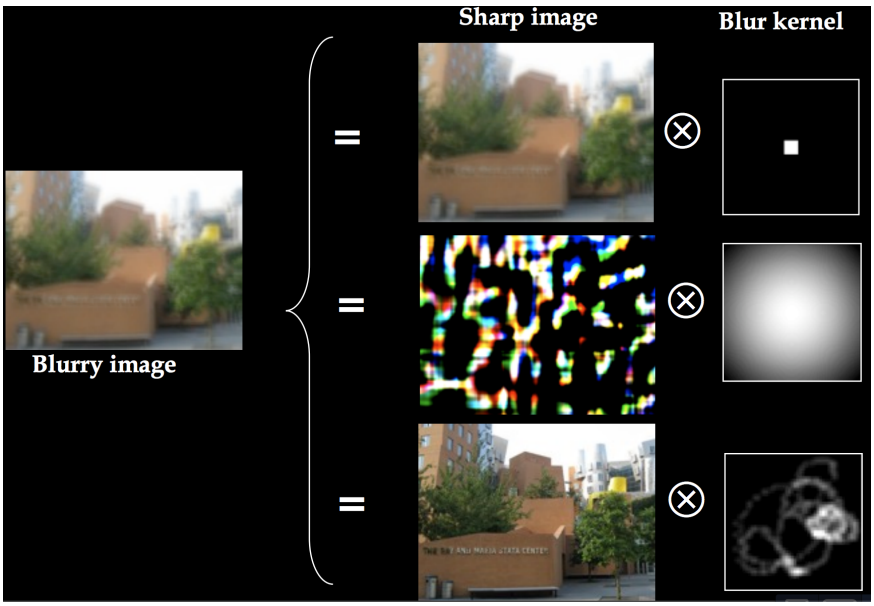

Figure 1: .....

从 Stata Center 的模糊照片找出清晰照片和模糊核的这个过程（特别对于计算机来说）就是非常 ambigous 的[1]。

所以么，还是让我们先抛开各种借口，回到问题本身。当然~~如同我的一贯作风~~这次的标题取得有点宏大，其实写这篇日志的主要目的不过是想吐槽一下上一次 6.438 课的一道作业题而已.....`=.=bb`

总而言之呢，像 Kmeans 之类的大部分经典的聚类算法，都是需要事先指定一个参数 K 作为类别数目的。但是很多时候这个 K 值并不是那

[1] 顺便这个 Slides 本身也是很有意思的，推荐看一下。

么好确定的，更麻烦的是，即使你使用很暴力的方法把某个范围内的整数 K 全部都试一遍，通常也没法知道哪个 K 才是最好的。

所以呢，我们自然会问：那有没有算法不需要指定类别数呢？当然有！最简单的就是层次聚类，它会将你的数据点用一个树形结构给连起来，可是我想要的是聚类结果啊！这也好办，只要在合适的树深度的地方把树切开，变成一个一个的子树，每个子树就是一个 cluster。不过问题就来了，究竟什么深度才是合适的深度呢？事实上不同的深度会产生不同的类别数目，这基本上把原来的选择 K 的问题转化成了选择树深度的问题。`^_^bb`

没关系，我们还有其他货，比如著名的 Mean Shift 算法，也是不需要指定类别数目就可以聚类的，而且聚类的结果也不是一棵树那种奇奇怪怪的东西。具体 Mean Shift 算法是什么样的今天就不在这里讲了，不过它其实也有一个参数 Window Size 需要选择——对，正如大家所料，这个 Window Size 其实也是会影响最终聚类的类别数目。换句话说，原来选择 K 的问题现在被转化为了选择 Window Size 的问题。

然后我们进入贝叶斯推断的领域，之前跟别人聊天的时候就听说过有个叫做 Dirichlet Process 的东西，特别神奇，能够自动地更具数据 adaptive 地确定合适的类别数目。我感觉这背后可能像其他算法一样会隐藏着其他的参数需要调节，不过鉴于我对 DP 完全不懂，就不在这里说胡话了，感兴趣的同学可以去研究一下。

然后贝叶斯推断聚类里的另一个成员就是 Affinity Propagation，其实是在讲了 Loopy Belief Propagation 算法之后的一道作业题，给了一个 factor graph，让把消息传递的公式推出来然后实现出来。看起来好像蛮容易的样子，实际上折腾了好几天，因为实现的时候碰到各种 tricky 的问题。

AP 算法把聚类问题看成一个 MAP 推断问题，假设我们有 n 个数据点 $p_1, \ldots, p_n$，这里我们要求类别中心实际上是这些数据点中的一个子集，聚类的结果可以用 $\{x_{ij}\}$ 这 n^2 个 binary variable 来表示，其中 $x_{ij} = 1$ 表示点 p_i 被归到点 p_j 所代表的那个类别中。

给定 n^2 个随机变量 $\{x_{ij}\}$ 之后，接下来是要通过 local factor 来定义他们的 (unnormalized) joint distribution。首先我们要求已知一个 Affinity Matrix S，其中 S_{ij} 表示点 p_i 和点 p_j 之间的相似程度[2]，一个简单的做法就是令

$$S_{ij} = -\|p_i - p_j\|$$

然后整个 joint distribution 主要由两类 factor 构成，在图 2 中给出了 3 个数据点时候的 factor graph 的例子。其中橙色的 factor 定义为

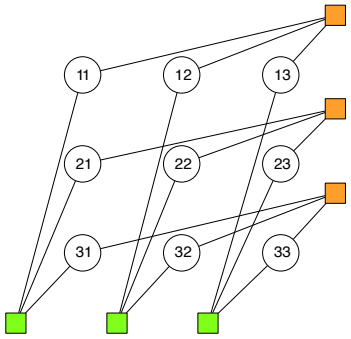

Figure 2: Factor graph demonstration for Affinity Propagation of 3 points.

[2] 这里 S 并不一定要求是对称的。

$$\phi_i(x_{i1},\dots,x_{in}) = \mathbb{1}\left(\sum_k x_{ik} = 1\right)\exp\left(\sum_k x_{ik}S_{ik}\right)$$

直观地来讲，该 factor 的第一部分是说每个点只能且必须被 assign 到一个 cluster；第二部分是讲 p_i 被 assign 到 p_j 的时候 factor 的值为 $e^{S_{ij}}$，也就是说会偏向于 assign 到相似度较大的那个 cluster 去。

而绿色的那些 factor 则定义为：

$$\psi_j(x_{1j},\dots,x_{nj}) = \mathbb{1}\left(x_{jj} \geq x_{ij}, i = 1,\dots,n\right)$$

这样的 indicator factor 要求如果有任何点 x_i 被 assign 到 x_j 的话，那么 x_j 必须要被 assign 到它自己。换句话说，每个 cluster 的中心必须是属于它自己那个类的，这也是非常合理的要求。

好了，模型的定义就是这么简单，这样我们会得到一个 loopy factor graph，接下来只要在这个 graph 上跑一下 Max-Product Algorithm 就可以得到最优的 $\{x_{ij}\}$，从而得到聚类结果了，并且都不用指定类别数目什么的，因为合适的类别数目可以从数据中自动地推断出来！

简直太神奇了，我都不敢相信这是真的！结果，果然童话里都是骗人的……因为隐藏在背后的还有许多细节需要处理。首先有一些 trick 来降低算法的复杂程度，比如说，由于所有的变量都是 binary 的，因此在消息传递的过程中我们可以只传递等于 1 和等于 0 两种状态时的消息差；然后由于数值下溢的原因，一般都会对所有的消息取 log，这样 Max-Product 算法会变成 Max-Sum（或者 Min-Sum）；然后有些消息是不做任何操作就直接 literally 传到接下来的节点的，这样的消息可以从中间步骤中去掉，最后化简之后的结果会只留下两种消息，在经典的 Affinity Propagation 算法中分别被成为 responsibility 消息和 availability 消息，可以有另外的 intuitive 的解释，具体可以参见 [Frey and Dueck, 2007]。

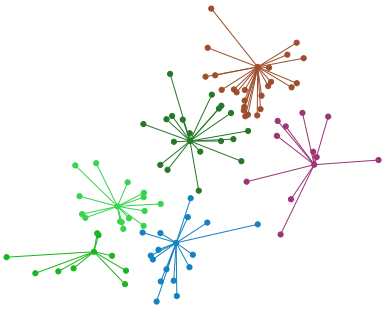

Figure 3: Clustering result of Affinity Propagation.

不过这些还是不太够，因为 Loopy BP 的收敛性其实是没有什么保证的，随便实现出来很可能就不收敛。比如说，如果直接进行 parallel 地 message updating 的话，很可能就由于 update 的幅度过大导致收敛困难，因此需要用 dampen 的方法将 fixed-point 的式子

$$x = f(x)$$

改为

$$x = 0.5x + 0.5f(x)$$

然后用这个来做 updating。当然也可以用 0.5 以外的其他 dampen factor。不过这样还是不够的，反正我折腾了好久最后发现必须把 parallel

updating 改成 in-place updating 才能收敛，这样一来更新的幅度就更小了一些，不过 dampen 仍然是需要的。另外就是一同讨论的另外的同学还发现实现中消息更新的顺序也是极端重要……把顺序变一下就立马从完全不 work 变成结果超好了。-_-!! 于是突然想起来 CV 课上老师拿 Convolutional Network 开玩笑的时候说，这个算法效果超级好，但是就是太复杂了，在发明这个算法的那个 lab 所在地的方圆 50 米之外完全没人能成功训练 Convolutional Network……

此外收敛性的判断也是有各种方法，比如消息之差的绝对值、或者是连续多少轮迭代产生的聚类中心都没有发生变化等等。即使在收敛之后，如何确定 assignment 也并不是显而易见的。因为 Max-Product 算法得到的是一些 Max-Marginal，如果全局最优的 configuration 不是 unique 的话，general 地是不能直接局部地通过各自的 Max-Marginal 来确定全局 configuration 的，于是可能需要实现 back-tracking，总之就是动态规划的东西。不过也有简单的 heuristic 方法可以用，比如直接通过 x_{ii} 的 Max-Marginal 来确定点 p_i 到底是不是一个 cluster center，确定了 center 之后则再可以通过 message 或者直接根据 Affinity 矩阵来确定最终的类别 Assignment。图 3 展示了一个二维情况下 100 个点通过 AP 聚类的结果。

不过，你有没有发现一点不对劲的地方？一切似乎太完美了：什么参数都不用给，就能自动确定类别数？果然还是有问题的。事实上，如果你直接就去跑这个算法，肯定得不到图上的结果，而是会得到一个 n 个类别的聚类结果：所有的点都成为了中心并且被归类到了自己。这样的结果也是可以理解的，因为我们计算 Affinity 的方式（距离的相反数）导致每个点和自己的 Affinity 是最大的……所以呢，为了解决这个问题，我们需要调整一个参数，就是 S 矩阵的对角线上的值，这个值反应了每个点自己成为类别中心的偏好程度，对的，如大家所料，这个就是背后的隐藏参数，直接影响类别数目。

如图 4 和 5 分别是把对角线上的元素值设置得比较大和比较小的情况，可以看到前者产生了更多的类别，因为大家都比较倾向于让自己成为类别中心；而后者则只产生了两个类，因为大家都很不情愿……（我发现这种画一些线连到类别中心的 visualization 方法似乎会给人产生很强的暗示感觉，于是各种聚类结果都看起来好像很“对”的样子 ^_^bb）

所以呀，到最后好像还是竹篮打水一场空的样子，怎么绕怎么绕好像也绕不开类别数目的这个参数。不过好像也并不是完全没有任何进展的，因为虽然各种算法都或多或少地需要指定一些参数，但是这些参数从不同的角度来阐释对应的问题，比如说在 AP 中，一般可以直接将 S 的对角线元素设置为其他所以元素的中位数，通常就能得到比较合理的结果；有时候对于特定的实际问题来说，从某一个角度来进行参数选择可能会有比较直观的 heuristic 可以用呢。所以说在是否需要指定参数这个问题上，还是不用太钻牛角尖了啊。^_^

最后我还想提一下，最近看到的 lab 的两篇 NIPS 文章 [Canas et al., 2012,

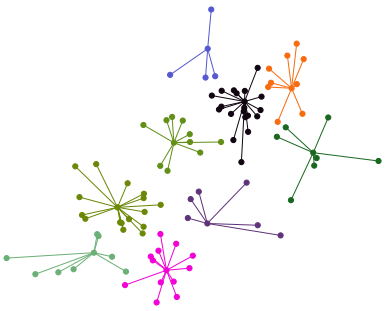

Figure 4: Clustering result of Affinity Propagation, with large diagonal values for the affinity matrix.

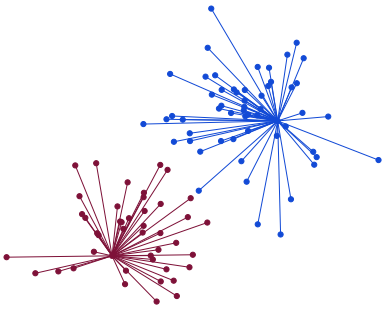

Figure 5: Clustering result of Affinity Propagation, with small diagonal values for the affinity matrix.

Canas and Rosasco, 2012]，里面的视角也是很有意思的。

我们知道，比如说，KMeans 算法的目标函数是这样定义的：

$$\mathcal{E}_n(S_K) = \frac{1}{n}\sum_{i=1}^{n} d^2(x_i, S_K)$$

其中 S_K 是一个由 K 个点组成的集合，而一个点 x_i 到 S_K 的距离被定义为到 S 中所有点的距离中最小的那一个值。这样可以看成是用 S_K 的这 K 个点去近似原本的 n 个点所带来的误差。然而这样的目标函数对于选择合适的 K 并没有什么指导意义，因为随着 K 的增大我们得到的最优的 S_K 会使得目标函数越来越小。

但是我们可以将这个目标函数推广一下，类似于 Supervised Learning Theory 中那样引入概率空间。具体地来说，我们假设数据点 $x_1, \ldots, x_n$ 是从一个未知的概率分布 ρ 中采样而得到的，并且目标函数也从对 n 个数据点的近似推广到整个生成流形的近似：

$$\mathcal{E}(S_K) = \int_{\mathcal{M}} d^2(x, S_K) d\rho(x)$$

当然由于 ρ 是未知的，所以 $\mathcal{E}(S_K)$ 是没法计算的，但是我们可以从 data sample 得到的 $\mathcal{E}_n(S_K)$ 来对它进行近似，并得到 bound 之类的。直观地来讲，此时对数值 K 的选择将会影响到模型空间的复杂度，于是会出现一个 trade-off，于是从这个角度下去探讨 “最优的 K 值” 就变成一个很合理的问题了。感兴趣的同学可以详细参考 paper 以及里面的参考文献，都在 arXiv 上可以下载到的。

References

[Canas et al., 2012] Canas, G. D., Poggio, T., and Rosasco, L. (2012). Learning manifolds with k-means and k-flats. In *NIPS*.

[Canas and Rosasco, 2012] Canas, G. D. and Rosasco, L. (2012). Learning probability measures with respect to optimal transport metrics. In *NIPS*.

[Frey and Dueck, 2007] Frey, B. J. and Dueck, D. (2007). Clustering by passing messages between data points. *Science*, 315:972–977.