manifold learning

Isomap

isomap 目标是在变换前后保持任意两点间的距离不变。

1.首先为每个样本集 X 中每个样本点 x 找到 K 个近邻点，并计算 x 到这个 K 个近邻的距离。

2.然后构建一幅图，每个样本点对应一个顶点，如果 x1 是 x2 的近邻，则 x1 和 x2 之间存在一条边，边的权重为 dist(x1,x2)。然后基于此图计算每两个点之间的距离，得到一个距离矩阵 D，这个距离就叫流形上的测地距离。

3.接下就是为每个 x 找一个低维表示 y，使得对任意的 x1 和 x2，有 dist(x1,x2)=dist(y1,y2)。

对 D 施以如下 cMDS 操作：

classical Multidimensional Scaling

input: $D \in \mathbb{R}^{n\times n}(D_{ii}=0,\ D_{ij}\geq 0),\ d\in\{1,\ldots,n\}$

1. Set $B := -\frac{1}{2}HDH$, where $H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ is the centering matrix.
2. Compute the spectral decomposition of B: $B = U\Lambda U^\top$.
3. Form Λ_+ by setting $[\Lambda_+]_{ij} := \max\{\Lambda_{ij}, 0\}$.
4. Set $X := U\Lambda_+^{1/2}$.
5. Return $[X]_{n\times d}$.

cMDS 有一个很重要的定理：

Theorem 1. *A nonnegative symmetric matrix $D \in \mathbb{R}^{n\times n}$, with zeros on the diagonal, is a Euclidean distance matrix if and only if $B \stackrel{\text{def}}{=} -\frac{1}{2}HDH$, where $H \stackrel{\text{def}}{=} I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$, is positive semidefinite. Furthermore, this B will be the Gram matrix for a mean-centered configuration with interpoint distances given by D.*

即由距离矩阵 D 算出的 B 是一个样本集 Y 的 gram 矩阵，即 B=YY^T，而这个 Y 就是 X 对应的低维表示。可以看出 B 就是 Y 的协方差矩阵(Y 的均值为 0)。

注意：k 个向量之间两两的内积所组成的矩阵，称为 Gram 矩阵。

Locally Linear Embedding(LLE)

LLE 假设每个点的邻域都是一个线性小片，也就是每个点可以被其 k 近邻的线性组合表示，他目标是在变换前后保持每个点与其 K 近邻之间的线性组合关系不变。

1.对每个点 x_i，求解下面的问题：

$$\begin{aligned}\min_{W_{ij}} \quad & \| x_i - \textstyle\sum_{j\in N(i)} W_{ij}x_j\| \\ s.t. \quad & \textstyle\sum_{j\in N(i)} W_{ij} = 1 \\ & W_{ij} = 0, \forall j \notin N(i)\end{aligned}$$

N(i)表示点 x_i 的 k 近邻集。

经过拉格朗日乘数法，这个问题最后等价于下面**|N(i)|*|N(i)|**的线性方程组：

$$GW_i = 1$$

其中：

$$G_{jk} = (x_i - x_j)^T(x_i - x_k)$$

2.接下来为每个 x 找一个低维表示 y，使得下式成立

$$y_i = \arg\min_{y_i} \quad \| y_i - \textstyle\sum_{j\in N(i)} W_{ij}y_j\|$$

所以在我们的低维空间中，每个点与其 k 近邻的线性组合关系不变。

这个问题可以重新写为：

$$\min_{y_{ij}} \quad \sum_i || \, y_i - \sum_j^N W_{ij} y_j ||$$

写为矩阵形式且加入我们两个约束，完整问题写为：

$$\begin{aligned} \min_{y_{ij}} \quad & Y^T (I-W)^T (I-W) Y \\ s.t. \quad & \sum_i^N Y_i = 0 \\ & Y^T Y = I \end{aligned}$$

约束 1 表示低维样本原点对称，注意这同时意味着这每个维度均值是 0。

约束 2 表示低维样本每个维度的方差为 1 且均值为 0。

需要特别注意一点：这个问题是 Rayleigh 商，他的解是(I-W)^T(I-W)的最小的 d 个特征值对应的特征向量，但是(I-W)^T(I-W)的最小的特征值及其对应的特征向量是(0,1)，这就导致 Y 的后面几个坐标取值是一样的全为 1。所以我们最后取的是(I-W)^T(I-W)最小的 d 个非 0 特征值对应的特征向量。

更细的过程可以参考下面的链接：

https://nbviewer.jupyter.org/github/drewwilimitis/Manifold-Learning/blob/master/Locally_Linear_Embedding.ipynb

SNE(Stochastic Neighbor Embedding)

[1]提出 SNE，在[2]中进行了再述。点 i 选择点 j 作为自己的近邻的概率为：

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)}$$

同理，对于低维空间的点 i 选择点 j 作为自己的近邻的概率则建模为：

$$q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|y_i - y_k\|^2\right)}$$

SNE 的目标是让低维分布去拟合高维分布，希望两个分布一致。定义代价函数:

$$C = \sum_i KL(P_i || Q_i) = \sum_i \sum_j p_{j|i} \, log \frac{p_{j|i}}{q_{j|i}}$$

注意，原来很远(p 大)但是映射后很近(q 小)的则惩罚大，反之惩罚小，所以 SNE 倾向去保留局部几何结构，不希望学习大尺度的结构结构。

固定一个点 i，其他点出现的概率分布是 P_i，定义每个 P_i 的 perplexity 值为：

$$perplexity(P_i) = 2^{H(p_i)}$$

其中：

$$H(p_i) = -\sum_j p_{(j|i)} \log_2 p_{(j|i)}$$

指定一个值 p，然后二分搜索找到一个\sigma_i 使得 perplexity 值等于 p。p 一般取值 5-50，每个点都取一样的 p，但是注意，每个点算出的\sigma_i 是不同的。

大约有 p 个 p_{j|i}不为 0，其他 p_{j|i}则非常接近 0(此话来自[3]的 Methods 一节).

对 C 进行梯度下降即可以学习到合适的 y_i。

$$\frac{\partial C}{\partial y_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

带动量的梯度更新公式：

$$y_i^{(t)} = y_i^{(t-1)} + \eta \frac{\partial C}{\partial y_i} + \alpha(t)(y_i^{(t-1)} - y_i^{(t-2)})$$

另外 Y 的初始值是从均值为 0，方差较小但各维度方差相同的高斯分布中随机采样得到的。

另外每轮迭代前 Y 被加了一个高斯噪声，噪声方差越来越小，这有点模拟退火的效果。

t-SNE(t-Distributed Stochastic Neighbor Embedding)

首先我们了解下 t 分布。t-distribution is a family of continuous probability distributions that arise when estimating the mean of a **normally-distributed** population in situations where the **sample size is small** and **the population's standard deviation is unknown**.

设 n 个 X_i 样本来自某个方差未知的正太分布$^{N(\mu,\sigma^2)}$，则样本均值和样本方差为：

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$$

而随机变量：

$$\frac{\bar{X} - \mu}{S/\sqrt{n}}$$

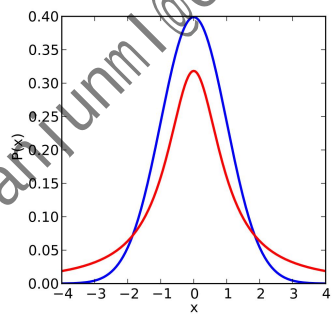
便服从自由度 v=n-1 的 t 分布。t 分布的 pdf 为：

$$f(t) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\,\Gamma(\frac{\nu}{2})}\left(1+\frac{t^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

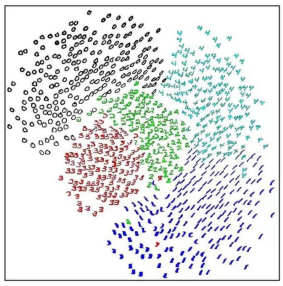
The t-distribution is symmetric and bell-shaped, like the normal distribution, but has heavier tails, meaning that it is more prone to producing values that fall far from its mean.

注意 t 分布的 pdf 中没有正态分布中的 exp 运算，gamma 函数在取定自由度后又是常数，所以计算速度比正太分布快。

t 分布尾巴比正太分布更高，所以适合长尾概率较高的场景。当 v=1 时等同柯西分布，当 v=inf 时等同标准正态分布，也即是 v 越大越接近正太分布，可以认为 v 代表了长尾强度。下图展示标准正态分布(蓝色)和自由度为 1 的 t 分布(红色)的 pdf 图像。



我们再来看看拥挤问题： 在低维维空间中，能容纳(高维空间中的)**中等距离**点对的空间更小。导致高维空间中离得较远的点，在低维空间中离得近，这就叫做拥挤问题。拥挤问题带来的一个直接后果，就是高维空间中分离的簇，在低维中被分的不明显。比如用 SNE 去可视化 MNIST 数据集的结果如下：



t-SNE 在高维空间使用高斯分布将距离转换为概率分布，在低维空间中使用更加偏重长尾的 t 分布来将距离转换为概率分布，**使得高维度空间中的中低等距离在映射后能够有一个较大的距离缓解拥挤问题。**

t-SNE 在高维空间中，使用联合概率来表示距离：

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

t-SNE 使用自由度为 1 的 t 分布替换高斯分布，低维空间中两点间的距离变为：

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}}$$

需要注意的一点是这里分母的下标是

$$\sum_{k=1,l=1,k!=l}^{K=n,l=n}$$

之所以这样是因为我们这里取的联合概率而非条件概率。

自由度为 1 的 t 分布的 pdf 为：

$$\frac{1}{\sqrt{\pi}}(1 + t^2)^{-1}$$

采用自由度为 1 的 t 分布是因为，对于低维空间中相距很远的点有：

$$\frac{1}{1 + ||y_i - y_j||^2} \approx \frac{1}{||y_i - y_j||^2}$$

此时这个距离是几乎缩放不变的(点都乘以某个常数后距离不变)。

然后让然优化 KL 散度：

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

得梯度如下：

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(1 + \|y_i - y_j\|^2)^{-1}(y_i - y_j)$$

具体推导过程见[2]的附录 A。

我们梳理下 t-SNE 的计算过程，如下：

a.先计算每个 p_ij，n^2 个实数，存储起来。

b.初始化全部 y_i，n*s 个实数，存储起来。

c.计算全部 q_ij，n^2 个实数，存储起来。

d.计算梯度。

e.采用梯度下降更新一把全部 y_i，goto c.

t-SNE 加速

[4]对 t-SNE 计算过程进行了加速。

首先因为对于绝大数多远离点 x_i 的 x_j 来说，p_{j|i}取值非常小，可以直接认为就是 0，所以 p_ij 的计算简化为：

$$p_{j|i} = \begin{cases} \frac{\exp(-d(\mathbf{x}_i, \mathbf{x}_j)^2 / 2\sigma_i^2)}{\sum_{k \in \mathcal{N}_i} \exp(-d(\mathbf{x}_i, \mathbf{x}_k)^2 / 2\sigma_i^2)}, & \text{if } j \in \mathcal{N}_i \\ 0, & \text{otherwise} \end{cases}$$

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}.$$

N_i 是 x_i 的 k 近邻集(k=ceil[3u]，u 就是指定的 perplexity 值)。而 k 近邻集的发现在利用 **vantage-point tree** 的情况下可以做到 O(uNlogN)的时间复杂度。

为了加快梯度的计算，首先把梯度重写为：

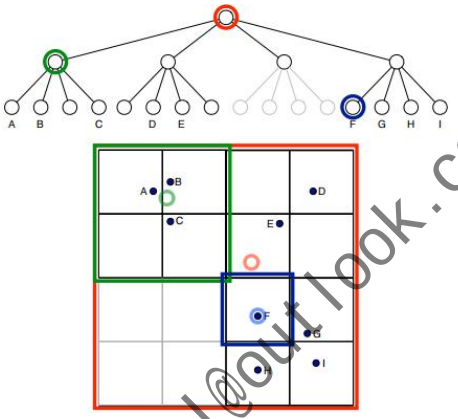
$$
\begin{aligned}
\frac{\partial C}{\partial y_i} &= 4\sum_j (p_{ij}-q_{ij})(1+||y_i-y_j||^2)^{-1}(y_i-y_j) \\
&= 4\sum_j (p_{ij}-q_{ij})q_{ij}\sum_{k\neq l}(1+||y_k-y_l||^2)(y_i-y_j) \quad \text{注意}q_{ij}\text{的计算公式} \\
&= 4\sum_j (p_{ij}-q_{ij})q_{ij}Z(y_i-y_j) \\
&= 4\left(\sum_j p_{ij}q_{ij}Z(y_i-y_j)-\sum_j q_{ij}^2 Z(y_i-y_j)\right) \\
&= 4\left(F_{attr}-F_{rep}\right) \quad F_{attr}\text{即上式的前半部分,}F_{rep}\text{则为后半部分}
\end{aligned}
$$

其中，Z 在每轮迭代前是一个常数，如下：

$$Z=\sum_{k\neq l}(1+\|\mathbf{y}_k-\mathbf{y}_l\|^2)^{-1}$$

因为很多 p_ij 都是 0，所以 F_attr 的计算会很快。

而计算 F_rep 前，我们先建立低维映射的四叉树。四叉树就是递归地对数据点所处的二维空间做四划分(中心点的西北，东北，西南，东南)形成的一棵树，根节点代表全部数据点分布空间，叶子节点代表至多一个数据点的分布空间。每个非叶子几点存储着本节点代表的空间的中样本点们的质心(y_cell)和样本点的总个数(N_cell)。插入是一个一个地插入到叶子节点，插入后叶子节点样个数大于 1 则分裂。如下：



我们将低维映射组织成一个四叉树后，如果一个节点足够小，且这个 cell 远离 y_i 时，那么这个 cell 里面的 y_j 差不多大小，我们就用其质心点代表 cell 里面的全部点，于是：

$$
\begin{aligned}
F_{rep} &= \sum_{cell}\sum_{j\in cell} q_{ij}^2 Z(y_i-y_j) \\
&\approx \sum_{cell} N_{cell} q_{i,cell}^2 Z(y-y_{cell}) \\
&= \sum_{cell} N_{cell} q_{i,cell}^2 Z^2(y-y_{cell}) * \tfrac{1}{Z} \\
&= \sum_{cell} N_{cell} (q_{i,cell}Z)^2 (y-y_{cell}) * \tfrac{1}{Z} \\
&= \sum_{cell} N_{cell} (1+||y_i-y_{cell}||^2)^{-2}(y-y_{cell}) * \tfrac{1}{Z}
\end{aligned}
$$

于是我们遍历四叉树，遇到一个节点(cell)，如果满足下式，则使用上述的粗略计算：

$$\frac{r_{cell}}{\|\mathbf{y}_i-\mathbf{y}_{cell}\|^2}<\theta,$$

r_cell 代表本 cell 对应的矩形的对角线长度,\thta 表示我们设定的误差。

四叉树中节点的子节点个数是 2^d，d 是维度大小，d 太大，则树就很大，一般好就 2-3 维。

参考文献

[1].Hinton, G., & Roweis, S. T. (2002, December). Stochastic neighbor embedding. In NIPS (Vol. 15, pp. 833-840).

[2].Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. Journal of machine learning research, 9(11).

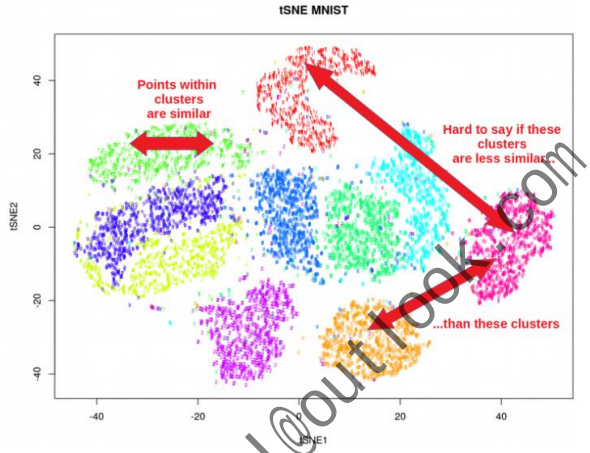
[3].Kobak, D., & Berens, P. (2019). The art of using t-SNE for single-cell transcriptomics. Nature communications, 10(1), 1-14.

[4].Van Der Maaten, L. (2014). Accelerating t-SNE using tree-based algorithms. The Journal of Machine Learning Research, 15(1), 3221-3245.

UMAP

tSNE 的缺点除了计算消耗和输出无法大于 3 维外，[1]的作者还提出下面几个：

- **tSNE can not work with high-dimensional data directly, Autoencoder or PCA are often used for performing a pre-dimensionality reduction before plugging it into the tSNE.**估计是因为高维空间欧式距离不可靠的原因。
- tSNE performs a non-parametric mapping from high to low dimensions, meaning that it does not leverage features that drive the observed clustering.这个确实，tSNE 只用到样本间距离信息。
- tSNE does not preserve global data structure, meaning that only within cluster distances are meaningful while between cluster similarities are not guaranteed, therefore it is widely acknowledged that clustering on tSNE is not a very good idea. 之前所有算法都是在追求保持局部结构的，没有说要保持全局结构。



[2]提出了 UMAP 算法。UMAP 的数学基础比 tSNE 更坚实，**涉及到黎曼几何，拓扑论等**，理解起来也蛮有难度。UMAP 也是先在高维空间中构建一个 k 邻图，然后基于某个目标函数学习一个低维表示，使得低维表示的 k 邻图尽可能保持某种高维 k 邻图的性质。

UMAP 先在高维空间中构建一个 k 邻图，其边的权重如下：

$$w((x_i, x_{i_j})) = \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right)$$

其中，d 是任意距离度量，k 是统一的邻居集大小，\rho_i 的定义为：

$$\rho_i = \min\{d(x_i, x_{i_j}) \mid 1 \le j \le k, d(x_i, x_{i_j}) > 0\}$$

，\sigma_i 则是满足下面方程的解：

$$\sum_{j=1}^{k} \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right) = \log_2(k)$$

然后把这个图的边经过如下变换得到一个新图 G，边权为：

$$P_{ij} = B_{ij} = W((x_i, x_j)) + W((x_j, x_i)) - W((x_i, x_j)) * W((x_j, x_i))$$

低维空间中两点的距离被定义为：

$$Q_{ij} = \Psi(\mathbf{y}_i, \mathbf{y}_j) = (1 + a||\mathbf{y}_i - \mathbf{y}_j)||^{2b})^{-1}$$

a，b 是两个参数，是下面的方程组的最小二乘解：

$$\Psi(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{if } \|\mathbf{x} - \mathbf{y}\|_2 \le \textit{min-dist} \\ \exp(-(\|\mathbf{x} - \mathbf{y}\|_2 - \textit{min-dist})) & \text{otherwise} \end{cases}$$

有默认值：

where a≈1.93 and b≈0.79 for default UMAP hyperparameters (in fact, for min_dist = 0.001).

优化的目标函数如下：

$$
\begin{aligned}
L &= \sum_i \sum_j P_{ij} \log \frac{P_{ij}}{Q_{ij}} + (1 - P_{ij}) \log \frac{1 - P_{ij}}{1 - Q_{ij}} \\
&= \sum_i \sum_j P_{ij} \log P_{ij} + (1 - P_{ij}) \log(1 - P_{ij}) - P_{ij} \log Q_{ij} - (1 - P_{ij}) \log(1 - Q_{ij}) \\
&= \sum_i \sum_j P_{ij} \log P_{ij} + (1 - P_{ij}) \log(1 - P_{ij}) - \sum_i \sum_j P_{ij} \log Q_{ij} + (1 - P_{ij}) \log(1 - Q_{ij}) \\
&= constant - \sum_i \sum_j P_{ij} \log Q_{ij} + (1 - P_{ij}) \log(1 - Q_{ij})
\end{aligned}
$$

用梯度下降法最小化这个目标函数时，求和符将带来很大的计算开销，所以作者使用采样方法来求解。具体的过程及其原理需进一步研究。另外，低维向量的初始化则采用所谓的 Spectral initialization 方法。

参考文献

[1]. How Exactly UMAP Works https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668

[2].McInnes, L., Healy, J., & Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426.