非线性降维方法——SNE, t-SNE, ClassNeRV

杨宇轩

2022.09.23

提　纲

01 非线性降维

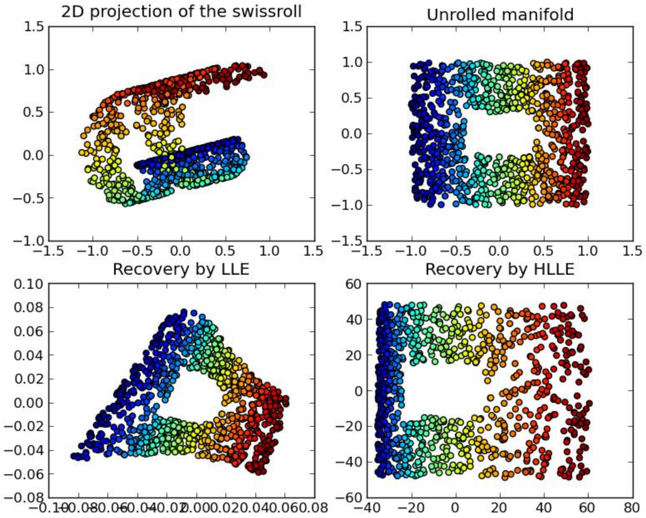
02 SNE

03 t-SNE

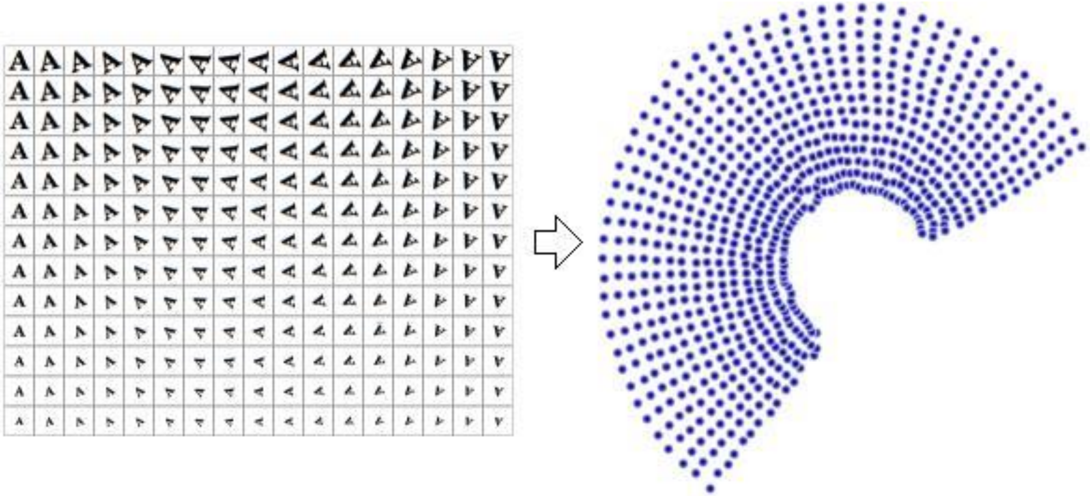
04 ClassNeRV

非线性降维

由于维度灾难、高维数据的难解释性等因素，我们需要用到降维方法，将数据从高维空间映射到低维空间。但以PCA为代表的线性降维方法表示能力不足，非线性降维则能够对高维数据进行更复杂的映射。主要分为两种：1. 提供一组映射的方法（可以进行样本外预测），2. 用于可视化（不能进行样本外预测）。



“瑞士卷”示例



左图：经旋转缩放变换的字母A图片数据，内禀维度为2，右图：降维结果

- 流形学习是非线性降维的子集

SNE

随机邻域嵌入（Stochastic Neighbor Embedding, SNE）是一种非线性降维方法，通常用于可视化，不能进行样本外预测。该方法的特点是将数据点映射到概率分布上，以保持数据点的邻域作为目标。方法主要分为三步：

1. 计算在原空间中数据点的邻域隶属度分布
2. 计算在嵌入空间中数据点的邻域隶属度分布
3. 优化两个概率分布之间的距离

Hinton G E, Roweis S. Stochastic neighbor embedding[J]. Advances in neural information processing systems, 2002, 15.

SNE

邻域隶属度条件概率

对于原空间数据点x_i和x_j，计算x_j属于x_i邻域的概率：

$$p_{j|i} = \frac{exp(-d_{j|i}^2/2\sigma_i^2)}{\sum_{k\neq i} exp(-d_{k|i}^2/2\sigma_i^2)}$$

- 将距离映射到高斯分布

- Softmax函数

其中$d_{j|i}$表示不相似度（距离），可以根据问题选定，SNE使用欧氏距离：

$$d_{j|i}^2 = \left\|x_i - x_j\right\|^2$$

由于只关注点对之间的相似度，令$p_{i|i} = 0$。

对于嵌入空间数据点y_i和y_j，可以类似地计算条件概率如下：

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

其中高斯分布的方差设为1/2（改变方差的值只会将结果同比例缩放），同样地，令$q_{i|i} = 0$。

SNE

σ_i的取值

σ_i决定了邻域的大小，通常来说，在密度较大的区域应取较小的σ_i值。对于x_i的邻域隶属度概率分布P_i（$p_{ij} \sim P_i$），分布的熵$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}$随$\sigma_i$值单调递增，SNE定义“困惑度（perplexity）”：

$$Perp(P_i) = 2^{H(P_i)}$$

困惑度的直观解释是x_i的有效邻域点数量，通常设为5~50之间。人为设定困惑度之后，σ_i的值可通过二分查找得到。

熵是整个系统的平均信息量，$2^{H(P_i)}$可以理解为系统中事件期望数量的度量。

1. 一篇文章中随机出现26个英文字母，熵$H = -\log_2 \frac{1}{26} = 4.7$，$2^{4.7} = 26$。
2. 随机变量X取三种可能值x_1, x_2, x_3，概率分别为$\frac{1}{2}, \frac{1}{4}, \frac{1}{4}$，熵$H = 1.5$，$2^{1.5} = 2.83$。

SNE

σ_i的取值

SNE

目标函数

为了使嵌入空间具有与原空间相同的邻域相似性，需要尽量使条件概率$p_{j|i}$和$q_{j|i}$相等，即让概率分布尽可能相匹配，因此可以使用KL散度作为目标函数：

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

其中P_i,Q_i分别表示x_i,y_i的邻域隶属度概率分布。

注意到KL散度具有不对称性，不同的类型的低维映射会带来不同的惩罚权重，具体来说是：当x_i,x_j近而y_i,y_j远时，$p_{i|j}$大，$q_{i|j}$小，KL散度计算结果较大；反之，当x_i,x_j远而y_i,y_j近时，$p_{i|j}$小，$q_{i|j}$大，KL散度计算结果较小，即：**SNE倾向于保留数据中的局部结构。**

SNE

优化

目标函数梯度为：

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

使用原点为中心，较小方差的高斯分布初始化Y。SNE的优化基于梯度下降，采用了以下trick：

1. 梯度下降时添加较大动量项。

2. 在较早的迭代中，每次在Y中添加高斯噪声，之后以模拟退火的方式减小噪声，以防止陷入局部最优。

在实际应用中，SNE的优化较为困难，一般需要多次调参。

t-SNE

SNE优化困难，而且存在“拥挤问题（crowding problem）”。针对这些问题，**t分布随机邻域嵌入（t-Distributed Stochastic Neighbor Embedding, t-SNE）**在SNE的基础上改进，主要的不同点在于：

1. 使用对称的邻域隶属度

2. 在低维空间中采用t分布计算邻域隶属度

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

t-SNE

原空间概率分布

SNE采用非对称的条件概率计算邻域隶属度，使得梯度较为复杂，优化困难。t-SNE采用**对称的联合概率分布**来解决这一问题。可以自然地对SNE公式进行扩展：

$$p_{ij} = \frac{exp(-\|x_i - x_j\|^2/2\sigma^2)}{\sum_{k\neq l} exp(-\|x_k - x_l\|^2/2\sigma^2)}$$

然而，上式会产生异常值的问题。假设x_i是异常值，那么$\|x_i - x_j\|^2$很大，对于$\forall j$，p_{ij}很小，导致y_i对loss影响小，y_i优化困难。

t-SNE采用另一种简单的计算方式：$p_{ij} = (p_{i|j} + p_{j|i})/2$，这保证了$\sum_j p_{ij} > 1/2n$，使得$y_i$对loss始终有一定影响。

t-SNE

对称SNE

采用对称概率，最大的好处是梯度变得简单了。令嵌入空间邻域隶属度为：

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

损失函数仍基于KL散度，条件概率分布改为联合概率分布：

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

则有如下梯度：

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)$$

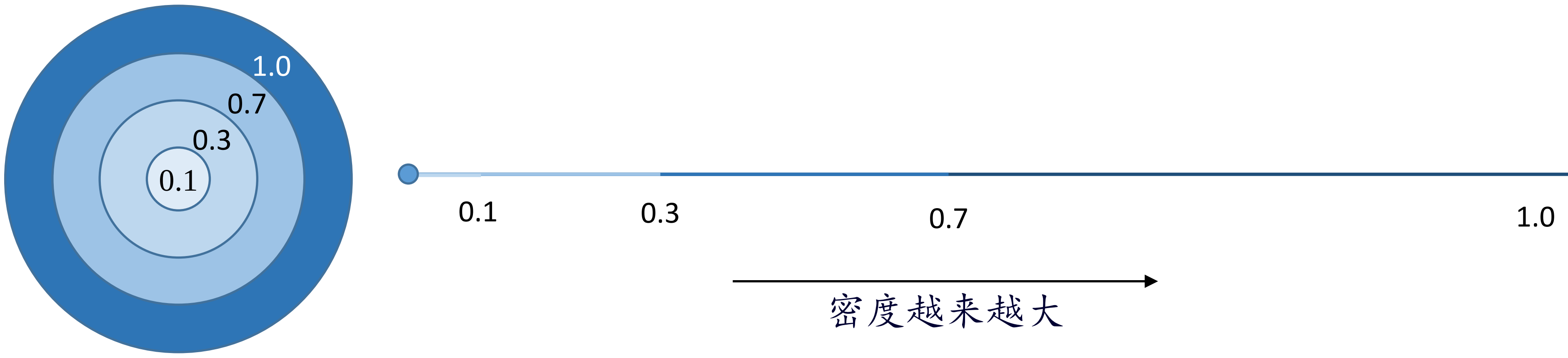
经过实验，对称SNE效果和原SNE差不多，甚至有时比原SNE更好。

t-SNE

拥挤问题（crowding problem）

一个特例：想象在10维空间中，存在11个点两两等距，那么当这11个点降维到2维空间中会发生什么——2维空间无法表示等距的11个点。低维空间里没有足够的位置去放高维空间中的点，即“拥挤问题”。

假设2维空间中有一个簇，以点i为中心，r为半径的球体内均匀分布，**密度**如下左图表示（原空间和嵌入空间邻域隶属度都采用高斯分布计算，因此可以忽略高斯分布的影响，直观考虑距离）。那么要降维到1维空间，同时保持邻域密度，对于点i为中心的密度图如右下所示，降维后的数据点聚集在外侧。



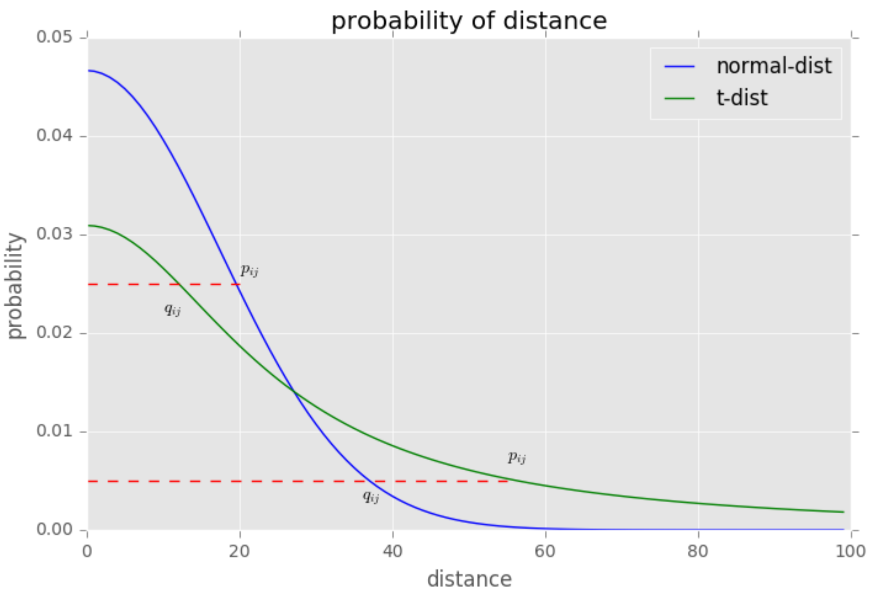
t-SNE

t分布

t-SNE将嵌入空间的邻域隶属度计算由高斯分布改为t分布，从而缓解拥挤问题。t分布的尾部比高斯分布更“重（heavy）”，概率随距离变化更慢，因此在同样的概率范围内能够放入更多的点，符合高维空间的分布。

t-SNE采用**自由度为1的t分布**，嵌入空间邻域隶属度为：

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



t-SNE

t-SNE梯度与优化

损失函数仍为：

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

梯度：

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

基于带动量的梯度下降进行优化，优化的trick：

1. 提前压缩（early compression）：初始化时，各个点要离得近一点。这样小的距离，方便各个聚类中心的移动。

2. 提前夸大（early exaggeration）：优化早期，p_{ij}乘以一个大于1的数，来避免p_{ij}太小导致优化太慢的问题。

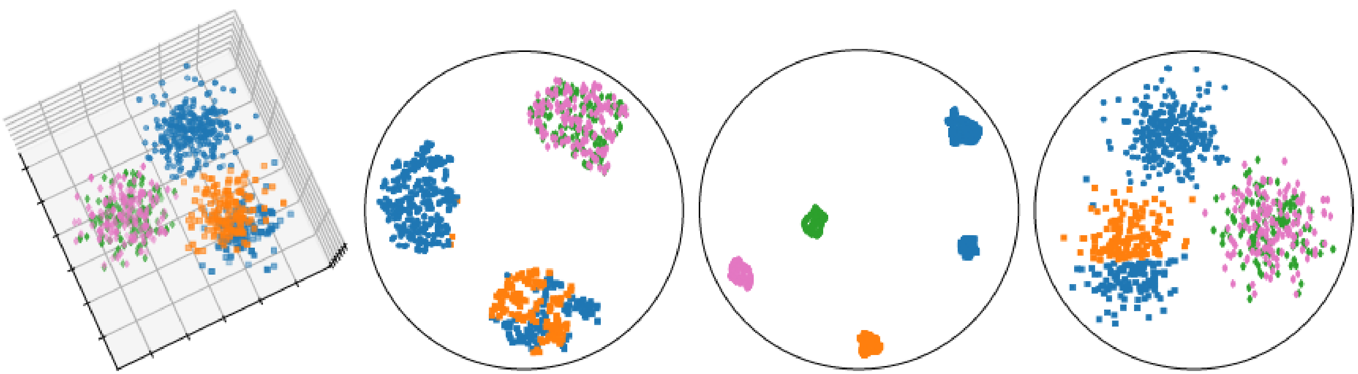
ClassNeRV

监督降维方法利用数据的相对位置和类标签计算降维映射，其中有两个相互矛盾的目标：

1. 分类是典型的监督式降维技术：强调类的分离，并用嵌入空间的分类精度来衡量。

2. 探索性数据分析是典型的无监督降维技术：优先考虑数据邻域结构，并以原始空间和嵌入空间的数据相似度之间的差异来衡量。

本文提出一种监督降维方法ClassNeRV，平衡了监督方法和无监督方法，具有类别判别性的同时也保留了邻域结构。



(a) 3D dataset (b) tSNE (unsupervised) (c) S-UMAP (supervised) (d) ClassNeRV

Colange B, Peltonen J, Aupetit M, et al. Steering distortions to preserve classes and neighbors in supervised dimensionality reduction[J]. Advances in neural information processing systems, 2020, 33: 13214-13225.

ClassNeRV

NeRV应力函数

NeRV是无监督的降维方法。令β_{ij}和b_{ij}分别表示在原空间和嵌入空间，点j隶属于点i邻域的概率，则有：

$$\beta_{ij}=\frac{exp(-\Delta_{\mathrm{ij}}^2/2\sigma_i^2)}{\sum_{k\neq i}exp(-\Delta_{\mathrm{ik}}^2/2\sigma_i^2)},b_{ij}=\frac{exp(-D_{ij}^2/2s_i^2)}{\sum_{k\neq i}exp(-D_{ik}^2/2s_i^2)}$$

在NeRV中，设定$s_i=\sigma_i$。NeRV应力函数（stress function）是两个KL散度的线性加权和：

$$\begin{aligned}\zeta_{NeRV}&=\sum_i \tau D_{Kl}(\beta_i,b_i)+(1-\tau)D_{KL}(b_i,\beta_i)\\&=\tau\sum_{i,j\neq i}\beta_{ij}log\left(\frac{\beta_{ij}}{b_{ij}}\right)+(1-\tau)\sum_{i,j\neq i}b_{ij}log(\frac{b_{ij}}{\beta_{ij}})\end{aligned}$$

其中，$\sum_i D_{KL}(\beta_i,b_i)$惩罚“missed neighbors”，$\sum_i D_{KL}(b_i,\beta_i)$惩罚“false neighbors”，$\tau\in[0,1]$控制权重。当$\tau=1$时，NeRV退化为对称SNE。

- missed neighbors：在原空间中是邻域点（近），但在嵌入空间中不是（远）
- false neighbors：在嵌入空间中是邻域点（近），但在原空间中不是（远）

ClassNeRV

ClassNeRV应力函数

在嵌入空间中，不应该把同一类别的数据点分离开，也不应该把不同类别的数据点聚在一起，因此，需要**更多地惩罚类内missed neighbors和类间false neighbors。**将NeRV应力函数中的散度拆成类内项和类间项，得到ClassNeRV应力函数如下：

$$\begin{aligned}
\zeta_{ClassNeRV} &= \sum_i \tau^{\in} D_B(\beta_i^{\in}, b_i^{\in}) + (1-\tau^{\in}) D_B(b_i^{\in}, \beta_i^{\in}) + \tau^{\notin} D_B(\beta_i^{\notin}, b_i^{\notin}) + (1-\tau^{\notin}) D_B(b_i^{\notin}, \beta_I^{\notin}) \\
&= \tau^{\in} \sum_{i,j \in S_i^{\in}} (\beta_{ij} \log(\frac{\beta_{ij}}{b_{ij}}) + b_{ij} - \beta_{ij}) + (1-\tau^{\in}) \sum_{i,j \in S_i^{\in}} (b_{ij} \log(\frac{b_{ij}}{\beta_{ij}}) + \beta_{ij} - b_{ij}) \\
&+ \tau^{\notin} \sum_{i,j \in S_i^{\notin}} (\beta_{ij} \log(\frac{\beta_{ij}}{b_{ij}}) + b_{ij} - \beta_{ij}) + (1-\tau^{\notin}) \sum_{i,j \in S_i^{\notin}} (b_{ij} \log(\frac{b_{ij}}{\beta_{ij}}) + \beta_{ij} - b_{ij})
\end{aligned}$$

其中，$S_i^{\in} = \{j \neq i | L_i = L_j\}$为类内集合，$S_i^{\notin} = \{j \neq i | L_i \neq L_j\}$为类间集合，$L_i$为点$i$的类别。注意到ClassNeRV用布雷格曼（Bregman）散度替换了KL散度。

ClassNeRV

布雷格曼散度

布雷格曼散度是一种通用的距离度量，这种距离满足：以任意概率分布取一系列点，这些点的平均值点一定是空间中距离这些点的平均距离最小的点。由函数$\varphi(x)$生成。

Table 1: Bregman divergences generated from some convex functions.

Domain	$\varphi(\mathbf{x})$	$d_\varphi(\mathbf{x},\mathbf{y})$	Divergence
$\mathbb{R}$	x^2	$(x-y)^2$	Squared loss
$\mathbb{R}_+$	$x\log x$	$x\log(\frac{x}{y})-(x-y)$	
$[0,1]$	$x\log x+(1-x)\log(1-x)$	$x\log(\frac{x}{y})+(1-x)\log(\frac{1-x}{1-y})$	Logistic loss [3]
$\mathbb{R}_{++}$	$-\log x$	$\frac{x}{y}-\log(\frac{x}{y})-1$	Itakura-Saito distance
$\mathbb{R}$	e^x	$e^x-e^y-(x-y)e^y$	
$\mathbb{R}^d$	$\|\mathbf{x}\|^2$	$\|\mathbf{x}-\mathbf{y}\|^2$	Squared Euclidean distance
$\mathbb{R}^d$	$\mathbf{x}^T A\mathbf{x}$	$(\mathbf{x}-\mathbf{y})^T A(\mathbf{x}-\mathbf{y})$	Mahalanobis distance [4]
d-Simplex	$\sum_{j=1}^d x_j\log_2 x_j$	$\sum_{j=1}^d x_j\log_2(\frac{x_j}{y_j})$	KL-divergence
$\mathbb{R}^d_+$	$\sum_{j=1}^d x_j\log x_j$	$\sum_{j=1}^d x_j\log(\frac{x_j}{y_j})-\sum_{j=1}^d(x_j-y_j)$	Generalized I-divergence

在ClassNeRV的应力函数中，隶属度概率的计算限制在$S_i^{\in},S_i^{\notin}$内，因此$\beta_i^{\in},b_i^{\in},\beta_i^{\notin},b_i^{\notin}$都不是和为1的概率分布，KL散度不再适用。

参考：https://www.zhihu.com/question/22426561，https://blog.csdn.net/wangshun_410/article/details/84963242

ClassNeRV

ClassNeRV应力函数

$$\zeta_{ClassNeRV} = \sum_i \tau^{\in} D_B(\beta_i^{\in}, b_i^{\in}) + (1-\tau^{\in}) D_B(b_i^{\in}, \beta_i^{\in}) + \tau^{\notin} D_B(\beta_i^{\notin}, b_i^{\notin}) + (1-\tau^{\notin}) D_B(b_i^{\notin}, \beta_I^{\notin})$$

$$= \tau^{\in} \sum_{i,j\in S_i^{\in}} (\beta_{ij}\log(\frac{\beta_{ij}}{b_{ij}}) + b_{ij} - \beta_{ij}) + (1-\tau^{\in}) \sum_{i,j\in S_i^{\in}} (b_{ij}\log(\frac{b_{ij}}{\beta_{ij}}) + \beta_{ij} - b_{ij})$$

$$+\tau^{\notin} \sum_{i,j\in S_i^{\notin}} (\beta_{ij}\log(\frac{\beta_{ij}}{b_{ij}}) + b_{ij} - \beta_{ij}) + (1-\tau^{\notin}) \sum_{i,j\in S_i^{\notin}} (b_{ij}\log(\frac{b_{ij}}{\beta_{ij}}) + \beta_{ij} - b_{ij})$$

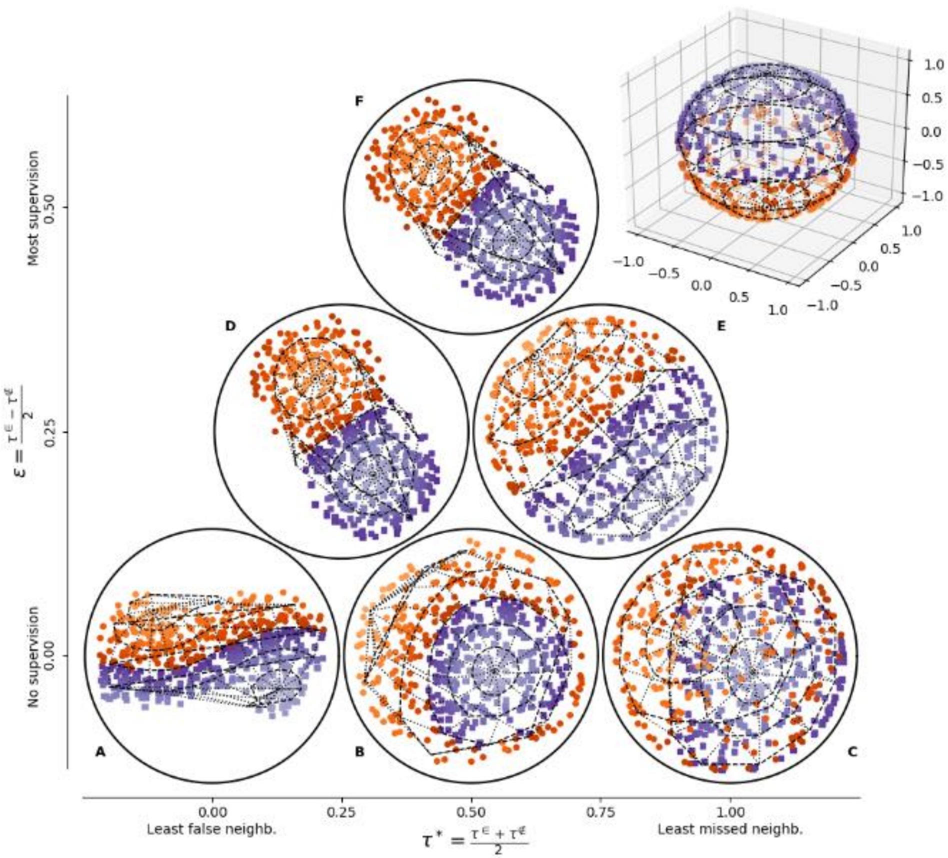
$D_B(\beta_i^{\in}, b_i^{\in}), D_B(b_i^{\in}, \beta_i^{\in}), D_B(\beta_i^{\notin}, b_i^{\notin}), D_B(b_i^{\notin}, \beta_i^{\notin})$这四项分别惩罚类内missed neighbors、类内false neighbors、类间missed neighbors、类间false neighbors。$\tau^{\in} \in [0,1]$控制类内missed neighbors和false neighbors的权重，$\tau^{\notin} \in [0,1]$控制类间两者的权重。

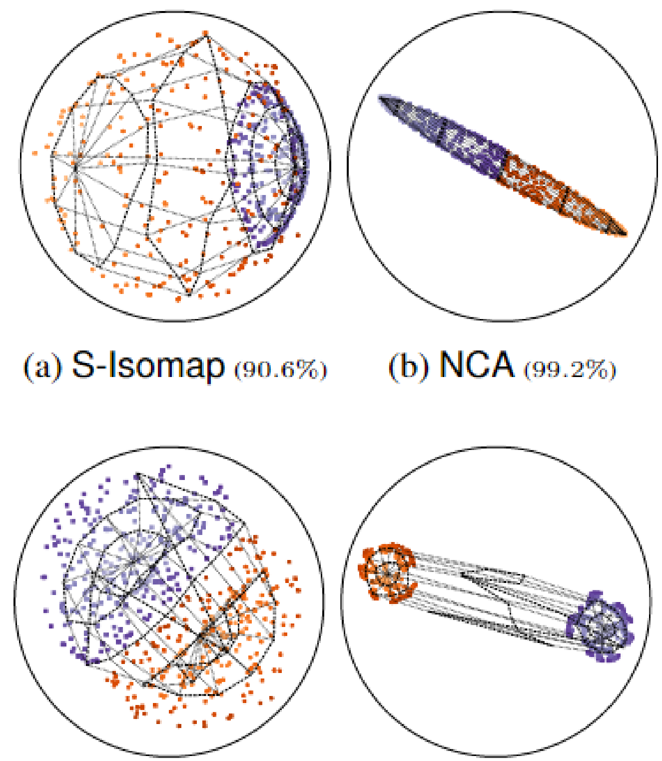
因此，**当$\tau^{\in} > \tau^{\notin}$，ClassNeRV是有监督的**，应力函数更多惩罚类内missed neighbors和类间false neighbors，当$\tau^{\in} < \tau^{\notin}$，应力函数鼓励了同类别数据的分离和不同类别数据的重叠，不利于类别监督。**当$\tau^{\in} = \tau^{\notin}$时，ClassNeRV是无监督的**，退化为原始的NeRV。

ClassNeRV

ClassNeRV应力函数

进一步令$\tau^* = (\tau^{\in} + \tau^{\notin})/2, \varepsilon = (\tau^{\in} - \tau^{\notin})/2$，$\tau^* \in [0,1]$控制missed neighbors和false neighbors的惩罚权重，$\varepsilon \in [0,0.5]$控制类别监督的水平（越大越有监督）。





(a) S-Isomap (90.6%) (b) NCA (99.2%)

(c) ClassiMap (99.8%) (d) S-UMAP (100%)

Q&A

合肥工業大學