高维数据可视化之 $t\text{-}SNE$ 算法

杨航锋

假设你有一个包含数百个特征的数据集，却对该数据所属领域几乎没有什么了解，并且你需要去探索数据中存在的隐模式。那可谓是数无形时少直觉，根本无从下手，当数据各特征间存在高度的线性相关，这时你可能首先会想到使用 PCA 对数据进行降维处理，但是 PCA 是一种线性算法，它不能解释特征之间的复杂多项式关系，而 $t\text{-}SNE$ (t-distributed stochastic neighbor embedding)是一种用于挖掘高维数据的非线性降维算法，它能够将多维数据映射到二维或三维空间中，**因此 $t\text{-}SNE$ 非常适用于高维数据的可视化操作。** $t\text{-}SNE$ 是由 SNE 发展而来，因此本文将先介绍 SNE 的基本原理，之后再扩展到 $t\text{-}SNE$ 。

1 SNE 基本原理

SNE 是通过仿射变换将数据点映射到相应概率分布上，主要包括下面两个步骤：

1. 通过在高维空间中构建数据点之间的概率分布 P ，使得相似的数据点有更高的概率被选择，而不相似的数据点有较低的概率被选择；
2. 然后在低维空间里重构这些点的概率分布 Q ，使得这两个概率分布尽可能相似。

令输入空间是 $X \in \mathbb{R}^n$ ，输出空间为 $Y \in \mathbb{R}^t$ $(t \ll n)$。不妨假设含有m个样本数据 $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$ ，其中 $x^{(i)} \in X$ ，降维后的数据为 $\{y^{(1)}, y^{(2)}, \cdots, y^{(m)}\}$ ， $y^{(i)} \in Y$ 。 SNE 是先将欧几里得距离转化为条件概率来表达点与点之间的相似度，即首先是计算条件概率 $p_{j|i}$ ，其正比于 $x^{(i)}$ 和 $x^{(j)}$ 之间的相似度， $p_{j|i}$ 的计算公式为：

$$p_{j|i} = \frac{exp(-\frac{\|x^{(i)} - x^{(j)}\|^2}{2\sigma_i^2})}{\sum\limits_{k \neq i} exp(-\frac{\|x^{(i)} - x^{(k)}\|^2}{2\sigma_i^2})}$$

在这里引入了一个参数 σ_i ，对于不同的数据点 $x^{(i)}$ 取值亦不相同，因为我们关注的是不同数据点两两之间的相似度，故可设置 $p_{i|i} = 0$ 。对于低维度下的数据点 $y^{(i)}$ ，通过条件概率 $q_{j|i}$ 来刻画 $y^{(i)}$ 与 $y^{(j)}$ 之间的相似度， $q_{j|i}$ 的计算公式为：

$$q_{j|i} = \frac{exp(-\|y^{(i)} - y^{(j)}\|^2)}{\sum\limits_{k \neq i} exp(-\|y^{(i)} - y^{(k)}\|^2)}$$

同理，设置 $q_{i|i} = 0$ 。

如果降维的效果比较好，局部特征保留完整，那么有 $p_{i|j}=q_{i|j}$ 成立，因此通过优化两个分布之间的 KL 散度构造出的损失函数为：

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

这里的 P_i 表示在给定高维数据点 $x^{(i)}$ 时，其他所有数据点的条件概率分布； Q_i 则表示在给定低维数据点 $y^{(i)}$ 时，其他所有数据点的条件概率分布。从损失函数可以看出，当 $p_{j|i}$ 较大 $q_{j|i}$ 较小时，惩罚较高；而 $p_{j|i}$ 较小 $q_{j|i}$ 较大时，惩罚较低。换句话说就是高维空间中两个数据点距离较近时，若映射到低维空间后距离较远，那么将得到一个很高的惩罚；反之，高维空间中两个数据点距离较远时，若映射到低维空间距离较近，将得到一个很低的惩罚值。也就是说， SNE **的损失函数更关注于局部特征，而忽视了全局结构。**

2 SNE 对应目标函数的求解

首先不同的数据点对应着不同的 σ_i ， P_i 的熵会随着 σ_i 的增加而增加。 SNE 引入困惑度的概念，通过二分搜索的方式来寻找一个最佳 $\boldsymbol{\sigma},\boldsymbol{\sigma}\in\mathbb{R}^n$ 。其中困惑度指：

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

这里的 $H(P_i)$ 是 P_i 的香农熵，即：

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

困惑度可以理解为某个数据点附近有效近邻点的个数， SNE **对困惑度的调整具有鲁棒性，通常选择** $5\sim50$ **之间**，给定之后通过二分搜索的方式即可寻找到合适的 $\boldsymbol{\sigma}$ 。通过对 SNE 的损失函数求梯度可得：

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

在迭代初始化阶段，可以使用较小的 $\boldsymbol{\sigma}$ 对高斯分布进行初始化。为了加速优化过程和避免陷入局部最优解，梯度中需要使用一个相对较大的动量，即参数更新过程中除了使用当前梯度，还要引入之前梯度累加的指数衰减项，迭代更新过程如下：

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

这里的 $Y^{(t)}$ 表示迭代 t 次的解， η 表示学习率， $\alpha(t)$ 表示迭代 t 次的动量。

3 对称 SNE

优化$KL(P||Q)$的一种替换思路是使用联合概率分布来替换条件概率分布，即 P 是高维空间里数据点的联合概率分布， Q 是低维空间里数据点的联合概率分布，此时的损失函数为：

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

同样的 $p_{ii} = q_{ii} = 0$ ，这种改进下的 SNE 称为对称 SNE ，因为它的先验假设为对 $\forall i$ 有 $p_{ij} = p_{ji}, q_{ij} = q_{ji}$ 成立，故概率分布可以改写成：

$$p_{ij} = \frac{exp(-\frac{\|x^{(i)} - x^{(j)}\|^2}{2\sigma^2})}{\sum_{k \neq l} exp(-\frac{\|x^{(k)} - x^{(l)}\|^2}{2\sigma^2})} \qquad q_{ij} = \frac{exp(-\|y^{(i)} - y^{(j)}\|^2)}{\sum_{k \neq l} exp(-\|y^{(k)} - y^{(l)}\|^2)}$$

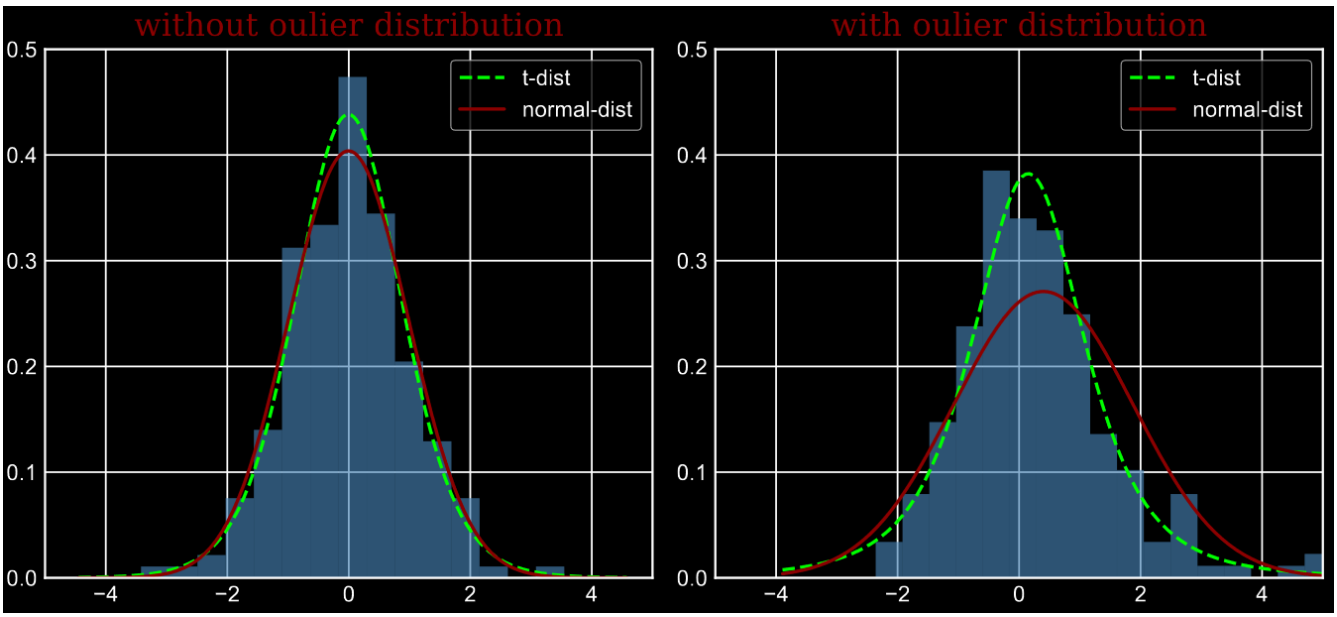
这种改进方法使得表达式简洁很多，但是容易受到异常点数据的影响，为了解决这个问题通过对联合概率分布定义修正为： $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2}$,这保证了 $\sum_j p_{ij} > \frac{1}{2m}$ ，使得每个点对于损失函数都会有贡献。对称 SNE 最大的优点是简化了梯度计算，梯度公式改写为：

$$\frac{\partial C(y^{(i)})}{\partial y^{(i)}} = 4 \sum_j (p_{ij} - q_{ij})(y^{(i)} - y^{(j)})$$

研究表明，对称 SNE 和 SNE 的效果差不多，有时甚至更好一点。

4 t-SNE

t-SNE 在对称 SNE 的改进是，首先通过在高维空间中使用高斯分布将距离转换为概率分布，然后在低维空间中，使用更加偏重长尾分布的方式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。



从图中可以看到，在没有异常点时，t 分布与高斯分布的拟合结果基本一致。而在第二张图中，出现了部分异常点，由于高斯分布的尾部较低，对异常点比较敏感，为了照顾这些异常点，高斯分布的拟合结果偏离了大多数样本所在位置，方差也较大。相比之下，t 分布的尾部较高，对异常点不敏感，保证了其鲁棒性，因此拟合结果更为合理，较好的捕获了数据的全局特征。

使用 t 分布替换高斯分布之后 q_{ij} 的变化如下：

$$q_{ij} = \frac{(1 + \|y^{(i)} - y^{(j)}\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y^{(i)} - y^{(j)}\|^2)^{-1}}$$

此外，随着自由度的逐渐增大，t 分布的密度函数逐渐接近标准正态分布，因此在计算梯度方面会简单很多，优化后的梯度公式如下：

$$\frac{\partial C(y^{(i)})}{\partial y^{(i)}} = 4\sum_j (p_{ij} - q_{ij})(y^{(i)} - y^{(j)})(1 + \|y^{(i)} - y^{(j)}\|^2)^{-1}$$

总的来说，$t\text{-}SNE$ 的梯度更新具有以下两个优势：

- 对于低维空间中不相似的数据点，用一个较小的距离会产生较大的梯度让这些数据点排斥开来；
- 这种排斥又不会无限大，因此避免了不相似的数据点距离太远。

5 与 PCA 降维效果对比

在 $MNIST$ 数据集上，该数据集的每张数字图片大小为 28×28 也就是说特征维度为 728，通过使用 $sklearn$ 算法库中的 $t\text{-}SNE$ 和 PCA 算法进行降维可视化测试，测试结果如下图所示：

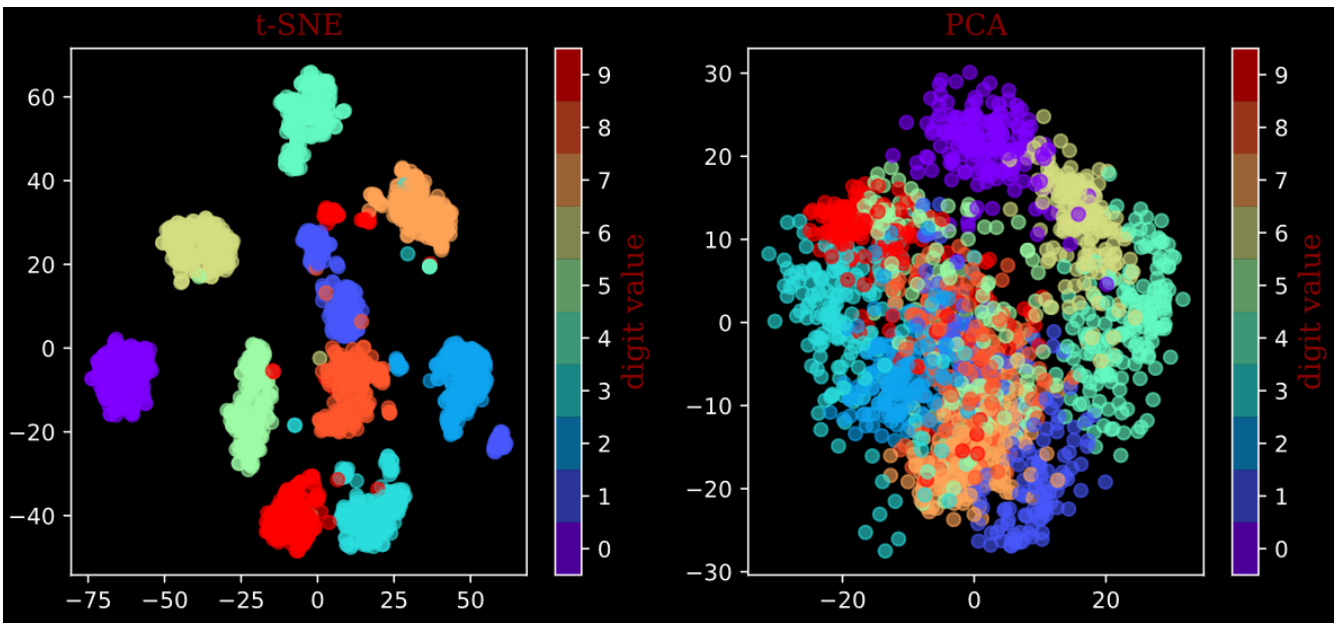
```
from sklearn.manifold import TSNE
from sklearn.datasets import load_iris,load_digits
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import numpy as np
%config InlineBackend.figure_format = "svg"

digits = load_digits()
X_tsne = TSNE(n_components=2, random_state=33).fit_transform(digits.data)
X_pca = PCA(n_components=2).fit_transform(digits.data)

font = {"color": "darkred",
        "size": 13,
```

```
        "family" : "serif"}

plt.style.use("dark_background")
plt.figure(figsize=(8.5, 4))
plt.subplot(1, 2, 1)
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=digits.target, alpha=0.6,
            cmap=plt.cm.get_cmap('rainbow', 10))
plt.title("t-SNE", fontdict=font)
cbar = plt.colorbar(ticks=range(10))
cbar.set_label(label='digit value', fontdict=font)
plt.clim(-0.5, 9.5)
plt.subplot(1, 2, 2)
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=digits.target, alpha=0.6,
            cmap=plt.cm.get_cmap('rainbow', 10))
plt.title("PCA", fontdict=font)
cbar = plt.colorbar(ticks=range(10))
cbar.set_label(label='digit value', fontdict=font)
plt.clim(-0.5, 9.5)
plt.tight_layout()
```



从实验结果可以看出 $t\text{-}SNE$ 算法降维后的可视化效果要远远好于 PCA 算法。

6 总结

$t\text{-}SNE$ 算法详细过程如下：

1. 数据准备： $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$ ， 其中 $x^{(i)} \in \mathbb{R}^n$ ；
2. 初始化困惑度参数用于求解 $\boldsymbol{\sigma}$ ， 迭代次数 T 、学习率 η 和动量 $\alpha(t)$ ；

3. 开始优化
    - 计算高维空间中的条件概率 $p_{j|i}$ ；
    - 令 $p_{ij}=\frac{p_{j|i}+p_{i|j}}{2m}$ ；
    - 使用正态分布 $\mathcal{N}(0,10^{-4})$ 随机初始化 $Y_{m\times k}$ 矩阵；
    - 从 $t=1,2,\cdots,T$ 进行迭代
        - 计算低维空间中的条件概率 q_{ij} ；
        - 计算损失函数 $C(y^{(i)})$ 对 $y^{(i)}$ 的梯度；
        - 更新 $Y^{(t)}=Y^{(t-1)}+\eta\frac{\partial C(Y)}{\partial Y}+\alpha(t)(Y^{(t-1)}-Y^{(t-2)})$ 。
    - 输出 Y 。
4. 算法结束。

t-SNE 算法其实就是在 SNE 算法的基础上增加了两个改进：

- 把 SNE 修正为对称 SNE ，提高了计算效率，效果稍有提升；
- 在低维空间中采用了 t 分布替换原来的高斯分布，解决了高维空间映射到低维空间所产生的拥挤问题，优化了 SNE 过于关注局部特征而忽略全局特征的问题 。