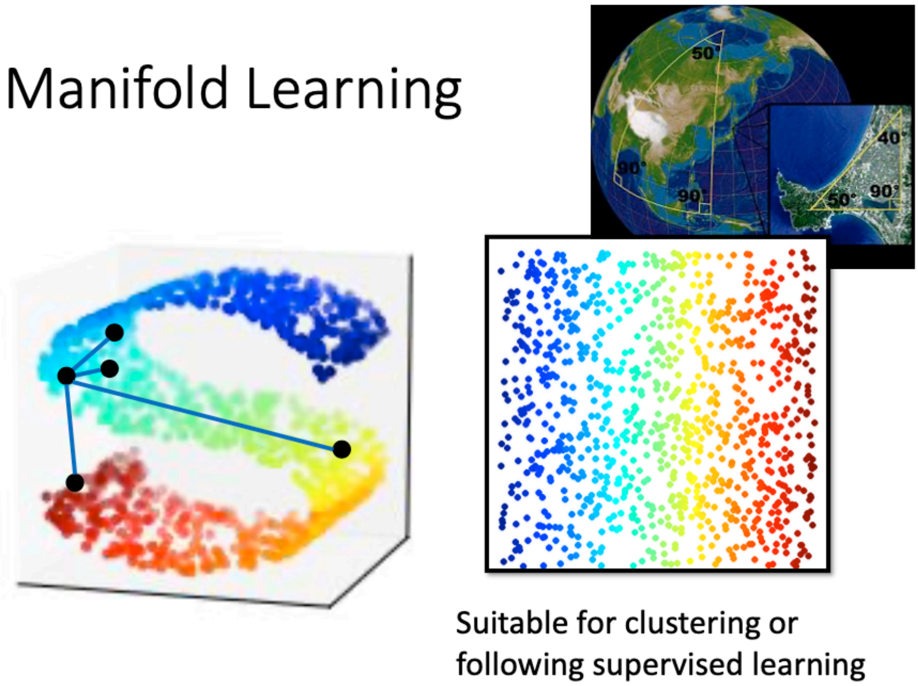
> 本文主要叙述了t-SNE，即T-distributed Stochastic Neighbor Embedding；先介绍了LLE的主要思想，再总结了它的缺点，从而引出t-SNE；

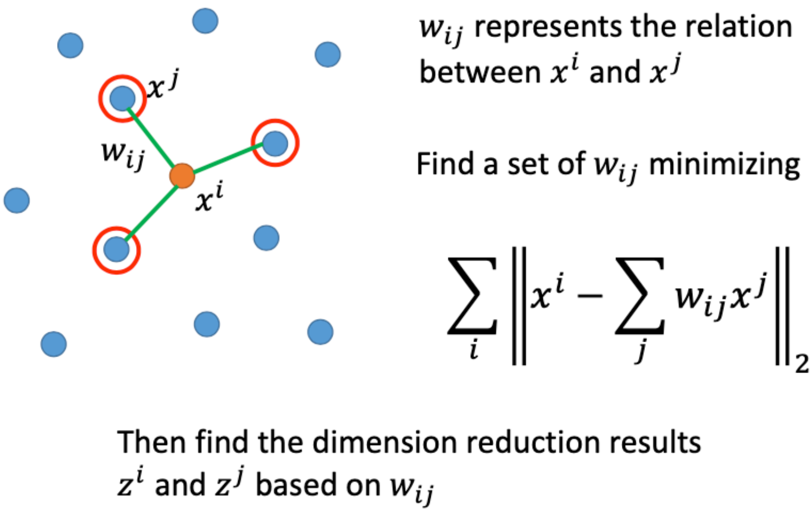
Manifold Learning

在高维空间里，距离该点很远的点很可能与这个点也是有关联的，因此我们可以把3-D的空间进行降维，那么我们就可以更方便地进行clustering或unsupervised learning 任务

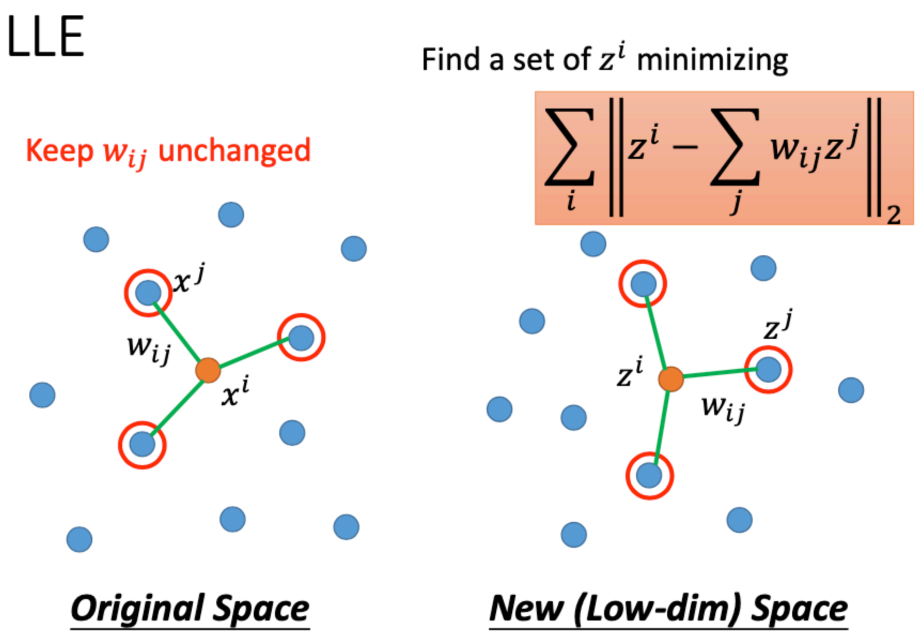


Locally Linear Embedding (LLE)

用w_{ij}表示x_i, x_j之间的联系，先找到使得$\sum_i ||x^i - \sum_j w_{ij}x^j||_2$最小化的$w_{ij}$，再根据这个$w_{ij}$来找到降维的结果$z_i, z_j$



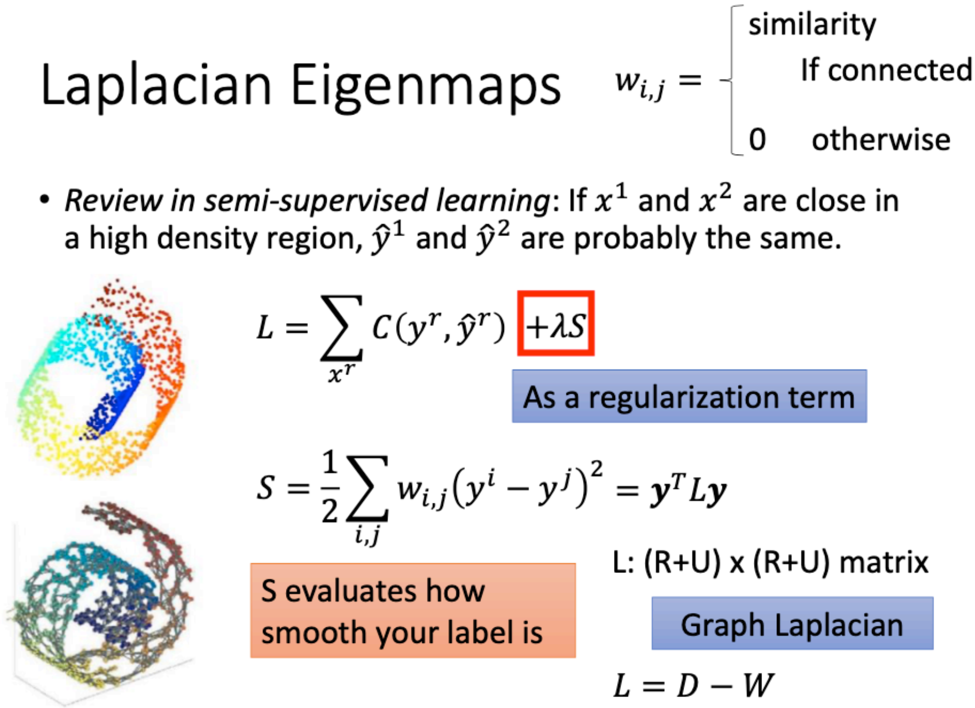
如果并不知道之前的x_i, x_j，那么就可以用LLE这种方法，也可以得出z_i, z_j



Laplacian Eigenmaps

Review: 在之前的semi-supervised learning中，如果x^1, x^2在一个high density region内是相近的，那么我们就可以认为$\hat{y}^1, \hat{y}^2$也有类似的表现

如果y^i, y^j是connected的，那么其w_{ij}就是对应的相似度；如果没有connect，其w_{ij}就是0



我们也可以得出类似smoothness的式子，计算z^i, z^j之间的smoothness

- *Dimension Reduction*: If x^1 and x^2 are close in a high density region, z^1 and z^2 are close to each other.

$$S = \frac{1}{2}\sum_{i,j} w_{i,j}\left(z^i - z^j\right)^2$$

Any problem? How about $z^i = z^j = \mathbf{0}$?

Giving some constraints to z:

If the dim of z is M, $\mathrm{Span}\{z^1, z^2, \ldots z^N\} = R^M$

Spectral clustering: clustering on z

Belkin, M., Niyogi, P. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems* . 2002

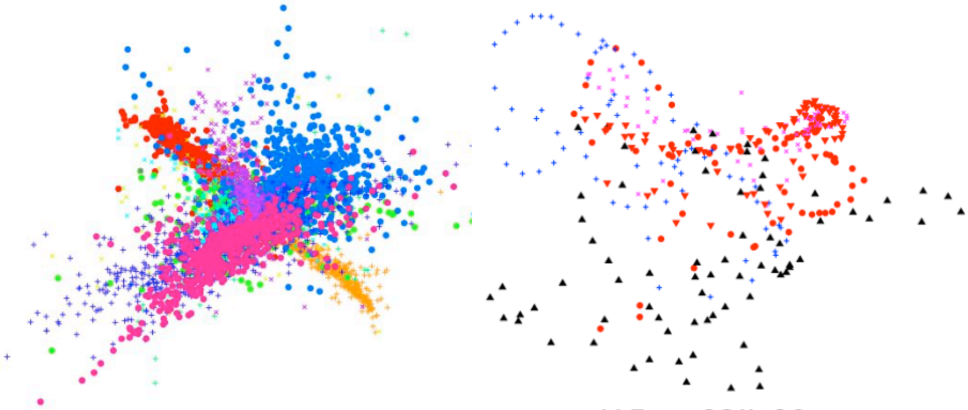
那么我们现在的目标就是找到z^i, z^j，来使S达到最小值，还需要有一些额外的constrains

现在我们对z加入一些constrains，如果降维后z的维数是M，那么我们就不希望取出来的这些点还生活在比M还低维的空间里面；我们现在希望把塞进高维空间的低维空间展开，我们就不希望展开之后的点在一个更低维的空间里面

T-distributed Stochastic Neighbor Embedding (t-SNE)

对于之前的LLE方法，类似的data之间是很close的，但不同类别之间的data却没有分开，是叠成一团的

- Problem of the previous approaches
  - Similar data are close, but different data may collapse



LLE on MNIST   LLE on COIL-20

为了找到对应的z^i, z^j，先计算x^i, x^j之间的相似度$S(x^i, x^j)$，再得出一个分布 $P(x^j|x^i)$；还要计算z^i, z^j之间的相似度$S'(z^i, z^j)$，得出分布$Q(z^j|z^i)$；

这两个分布应该越接近越好，使用L来表示

t-SNE

x ⟶ z

Compute similarity between all pairs of x: $S(x^i, x^j)$

Compute similarity between all pairs of z: $S'(z^i, z^j)$

$$P(x^j|x^i) = \frac{S(x^i, x^j)}{\sum_{k \neq i} S(x^i, x^k)}$$

$$Q(z^j|z^i) = \frac{S'(z^i, z^j)}{\sum_{k \neq i} S'(z^i, z^k)}$$

Find a set of z making the two distributions as close as possible

$$L = \sum_i KL\left(P(*|x^i)||Q(*|z^i)\right)$$

$$= \sum_i \sum_j P(x^j|x^i) log \frac{P(x^j|x^i)}{Q(z^j|z^i)}$$

可以使用gradient descent，有了L函数，再分别对z^i, z^j求偏微分即可

但t-SNE要对所有的point之间都求similarity，因此计算量比较大，在数据量很大的情况下电脑的计算速度会非常慢

因此，对于很高的dimensions，通常先做降维（PCA），比如可以降维到50维，再使用t-SNE降到2维

通常我们使用t-SNE来对高维的数据进行可视化

t-SNE –Similarity Measure

Ignore σ for simplicity

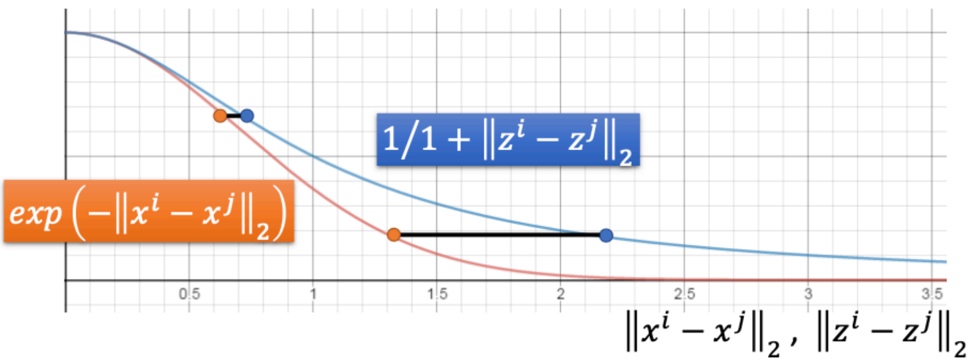
$$S(x^i, x^j) = exp\left(-\|x^i - x^j\|_2\right)$$

SNE:

$$S'(z^i, z^j) = exp\left(-\|z^i - z^j\|_2\right)$$

t-SNE:

$$S'(z^i, z^j) = 1/1 + \|z^i - z^j\|_2$$

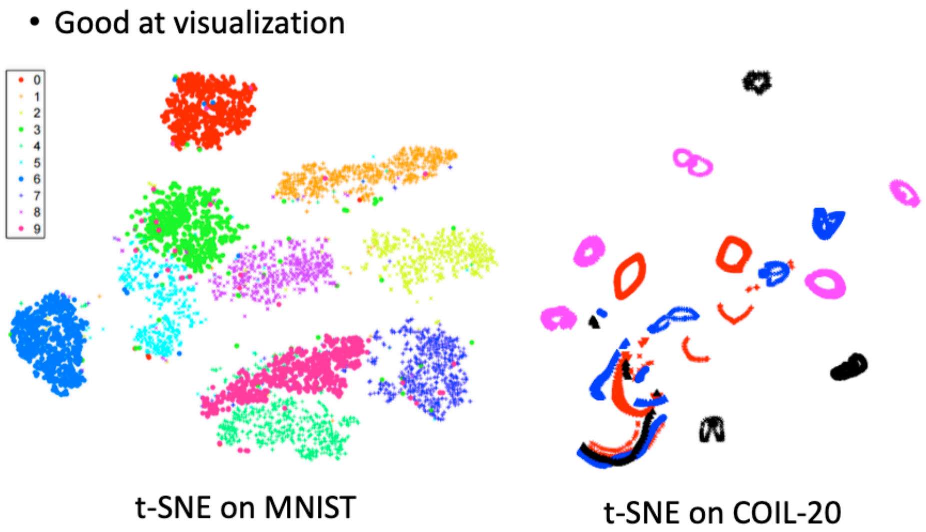


在上图中，红色曲线表示SNE，蓝色曲线表示t-SNE，纵轴表示distribution；如果我们想要维持相同的distribution，即在同一个水平线上，就达到了如图所示的效果；相同的几率，t-SNE的$||z^i - z^j||_2$之间的距离越大

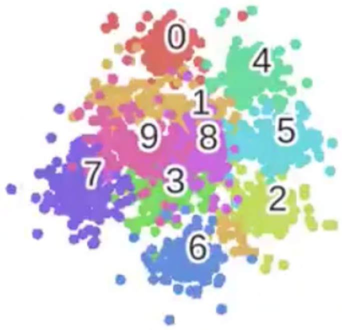
如果本来就离得很近，那么经过t-SNE之间的距离还是很小；如果本来就离得很远，那么从原来的distribution拉到t-SNE之后，距离会更远；

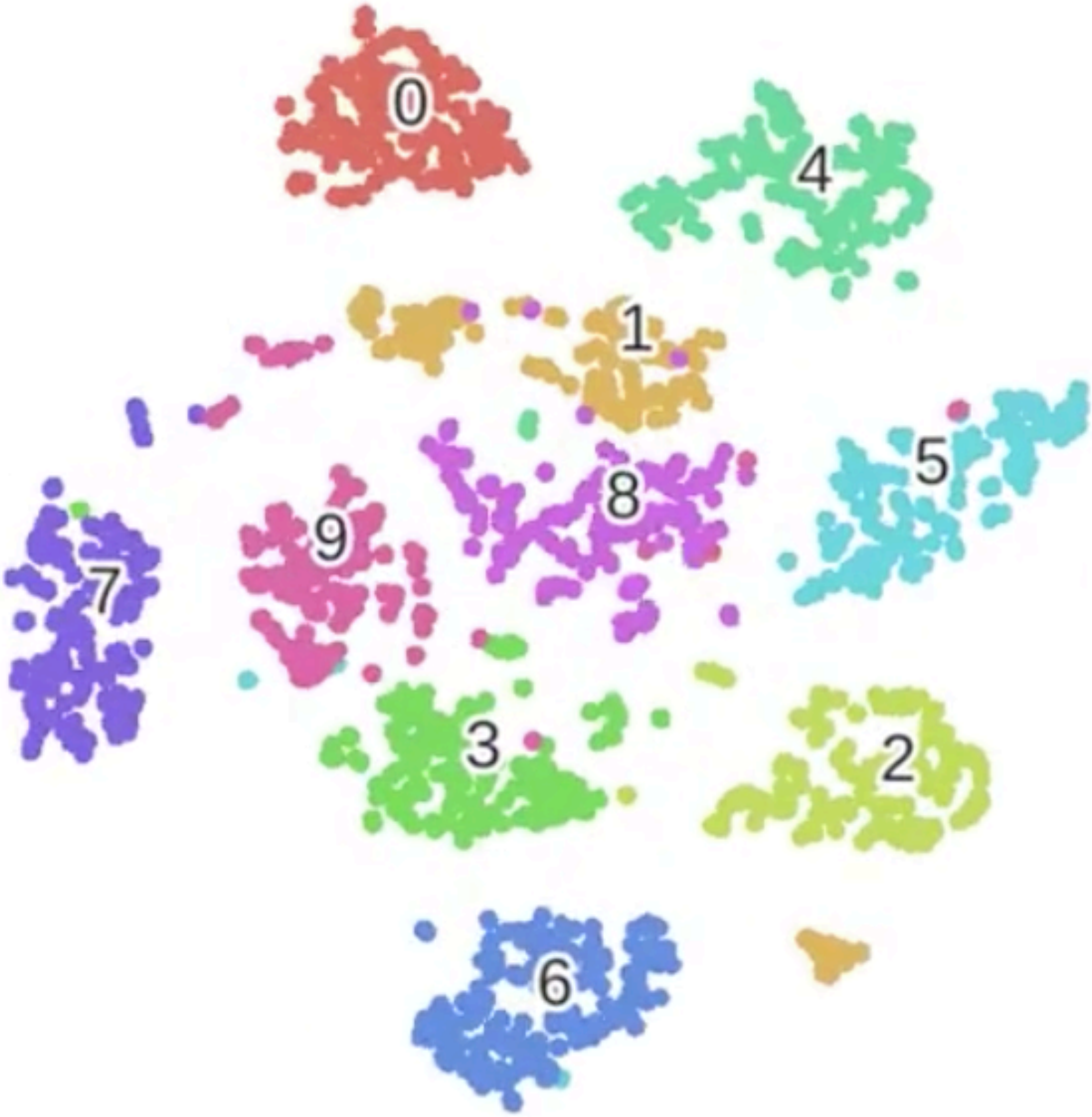
到实际的例子中，如果本来是同一个类别的data，由于这些data之间的距离很近，不会收到t-SNE很大的影响；但如果是属于不同类别的data，距离是比较远的，t-SNE会放大这种距离

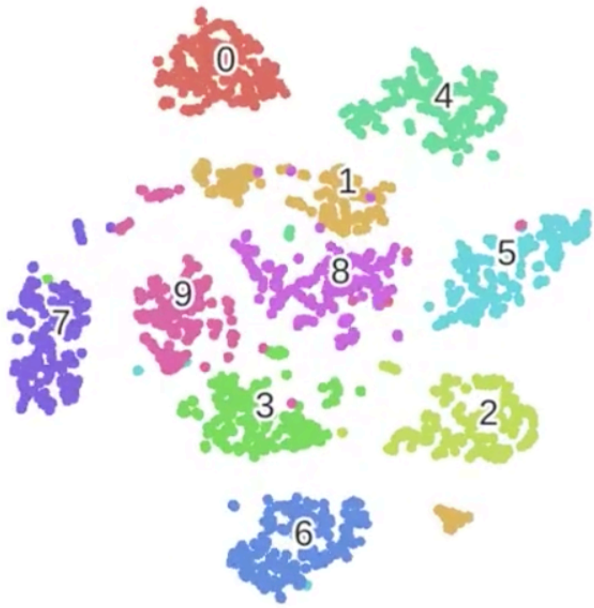
对于下图中的MNIST，先使用PCA进行降维，再进行可视化，就可以得到下图中的good visualization



下图中有一个更加直观的例子，使用t-SNE算法，运用gradient descent的思想，不同类别的data之间的距离会越来越大


0
4
1
9
8
5
7
3
2
6


0
4
1
5
8
9
7
3
2
6


0
4
1
5
8
9
7
3
2
6