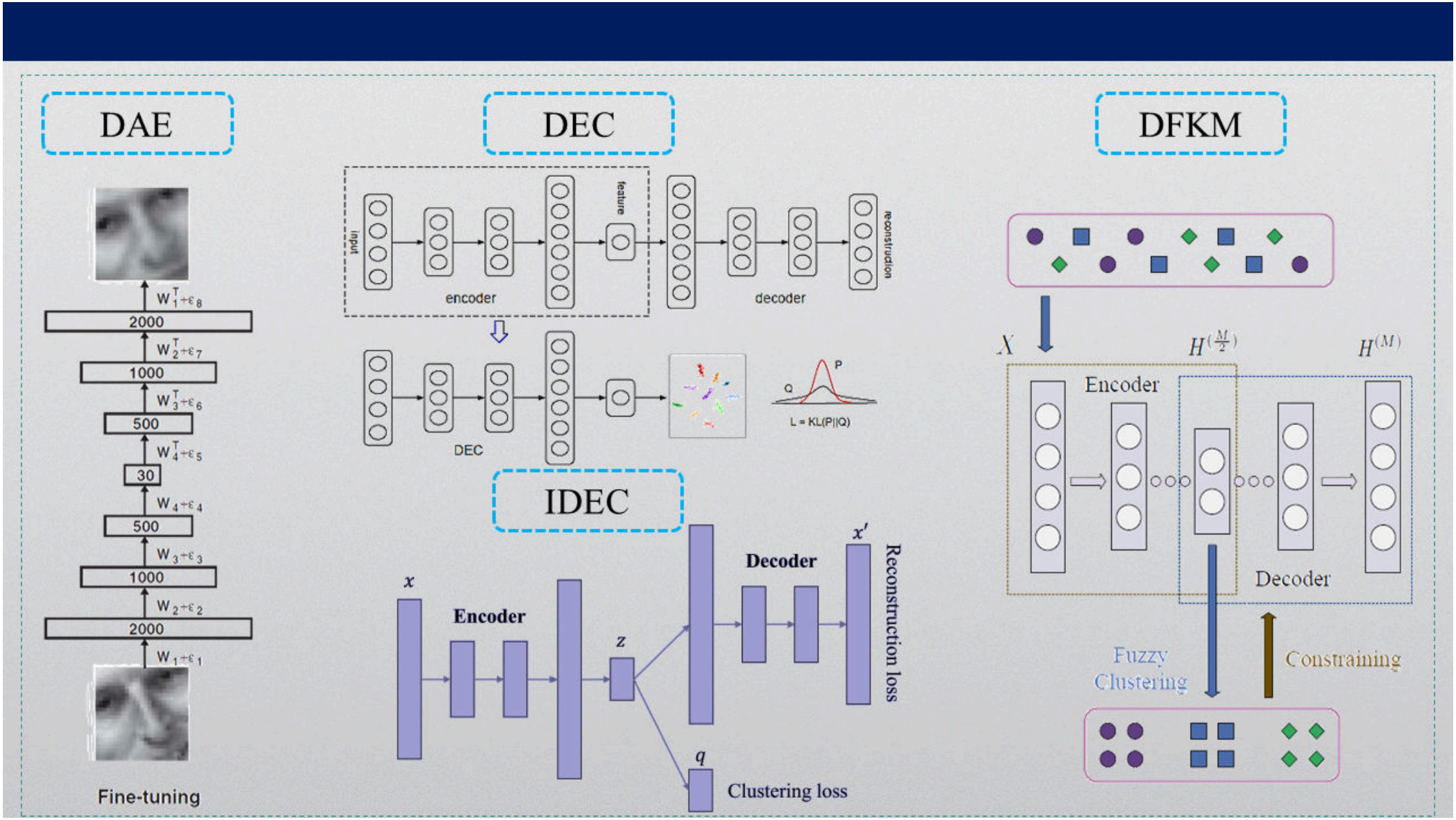
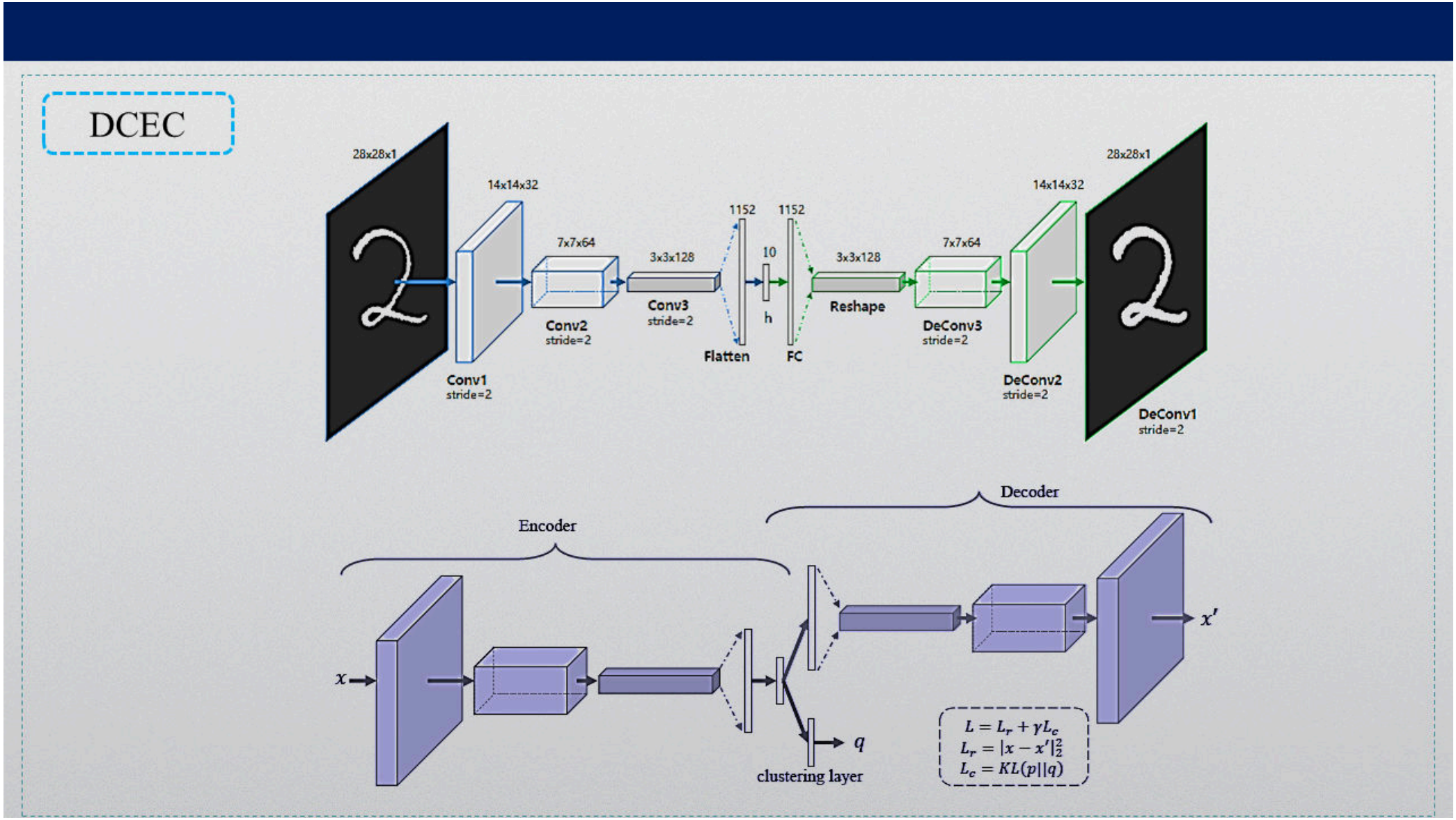
Deep Clustering Algorithms

作者：凯鲁嘎吉 - 博客园 http://www.cnblogs.com/kailugaji/

本文研究路线：深度自编码器(Deep Autoencoder)->Deep Embedded Clustering(DEC)->Improved Deep Embedded clustering(IDEC)->Deep Convolutional Embedded Clustering(DCEC)->Deep Fuzzy K-means(DFKM)，其中Deep Autoencoder已经在深度自编码器(Deep Autoencoder)MATLAB解读中提到，也有很多深度自编码器的改进方法，不详细讲解，重点谈深度聚类算法。如有不对之处，望指正。

深度聚类算法的网络架构图


DAE
DEC
DFKM
input
feature
reconstruction
encoder
decoder
DEC
P
Q
L = KL(P||Q)
IDEC
x
Encoder
z
Decoder
x′
Reconstruction loss
q
Clustering loss
2000
1000
500
30
500
1000
2000
Fine-tuning
X
Encoder
Decoder
Fuzzy Clustering
Constraining



深度聚类算法的损失函数

DAE

$$L = \| \mathbf{H}^{(M)} - \mathbf{X} \|_F^2$$

DEC

$$L = \sum_i^N \sum_j^k p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

IDEC

$$L = \| \mathbf{H}^{(M)} - \mathbf{X} \|_F^2 + \gamma \sum_i^N \sum_j^k p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

DCEC

$$L = \| \mathbf{H}^{(M)} - \mathbf{X} \|_F^2 + \gamma \sum_i^N \sum_j^k p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

DFKM

$$L = \| \mathbf{H}^{(M)} - \mathbf{X} \|_F^2 + \lambda_1 \sum_{i=1}^N \sum_{j=1}^k u_{ij} \| \mathbf{h}_i^{(\frac{M}{2})} - \mathbf{c}_j \|_{\hat{\sigma}} + \gamma u_{ij} \log u_{ij}$$

$$+ \lambda_2 \sum_{m=1}^M \| \mathbf{W}^{(m)} \|_F^2 + \| \mathbf{b}^{(m)} \|_2^2$$

1. Deep Embedded Clustering

1.1 Stochastic Neighbor Embedding (SNE)

SNE是一种非线性降维策略，两个特征之间存在非线性相关性，主要用于数据可视化，PCA（主成成分分析）是一种线性降维策略，两个特征之间存在线性相关性。SNE在原始空间(高维空间)中利用Gauss分布将数据点之间的距离度量转化为条件概率，在映射空间(低维空间)中利用Gauss分布将映射点之间的距离度量转化为条件概率，并利用KL散度来最小化高维空间与低维空间的条件概率。

➢ Stochastic Neighbor Embedding (SNE)

在高维空间相似的数据点，映射到低维空间距离也是相似的。常规的做法是用欧式距离表示这种相似性，而SNE把这种距离关系转换为一种条件概率来表示相似性。考虑高维空间中的两个数据点x_i和x_j，x_i以条件概率$p_{j|i}$选择x_j作为它的邻近点。考虑以x_i为中心点的高斯分布，若x_j越靠近x_i，则$p_{j|i}$越大。反之，若两者相距较远，则$p_{j|i}$极小。因此，我们可以这样定义$p_{j|i}$：

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

当我们把数据映射到低维空间后，高维数据点之间的相似性也应该在低维空间的数据点上体现出来。这里同样用条件概率的形式描述，假设高维数据点x_i和x_j在低维空间的映射点分别为y_i和y_j。类似的，低维空间中的条件概率用$q_{j|i}$表示，并将所有高斯分布的方差均设定为$\frac{1}{\sqrt{2}}$，所以有：

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

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

Maaten L, Hinton G. Visualizing data using t-SNE[J]. Journal of machine learning research, 2008, 9(Nov): 2579-2605.

SNE面临的问题有两个：（1）KL散度是一种非对称度量，（2）拥挤问题。对于非对称问题，定义pij，将非对称度量转化为对称度量。但对称度量仍然面临拥挤问题，映射到低维空间中，映射点之间不能根据数据本身的特性很好地分开。

➢ Stochastic Neighbor Embedding (SNE)

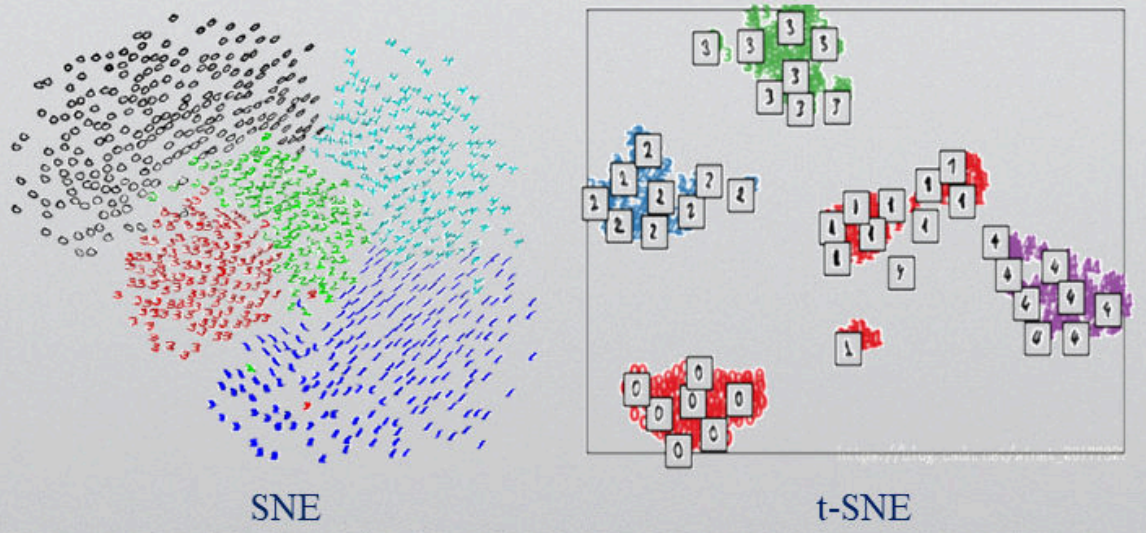
KL距离是一个非对称的度量。最小化代价函数的目的是让$p_{j|i}$和$q_{j|i}$的值尽可能的接近，即低维空间中点的相似性应当与高维空间中点的相似性一致。但是从代价函数的形式就可以看出，当$p_{j|i}$较大，$q_{j|i}$较小时，代价较高；而$p_{j|i}$较小，$q_{j|i}$较大时，代价较低。很显然，高维空间中两个数据点距离较近时，若映射到低维空间后距离较远，那么将得到一个很高的惩罚，这当然没问题。反之，高维空间中两个数据点距离较远时，若映射到低维空间距离较近，将得到一个很低的惩罚值，这就有问题了，理应得到一个较高的惩罚才对。换句话说，SNE的代价函数更关注局部结构，而忽视了全局结构。

$$q_{ij}=\frac{\exp(-\|y_i-y_j\|^2)}{\sum_{k\neq i}\exp(-\|y_i-y_k\|^2)} \qquad p_{ij}=\frac{\exp(-\|x_i-x_j\|^2/2\sigma^2)}{\sum_{k\neq i}\exp(-\|x_i-x_k\|^2/2\sigma^2)}$$

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

$$L=\sum_i^N KL(P_i\|Q_i)=\sum_i^N\sum_j^k p_{ij}\log\frac{p_{ij}}{q_{ij}}$$

拥挤问题(The Crowding Problem)。拥挤问题的出现与某个特定算法无关，而是由于高维空间距离分布和低维空间距离分布的差异造成的。

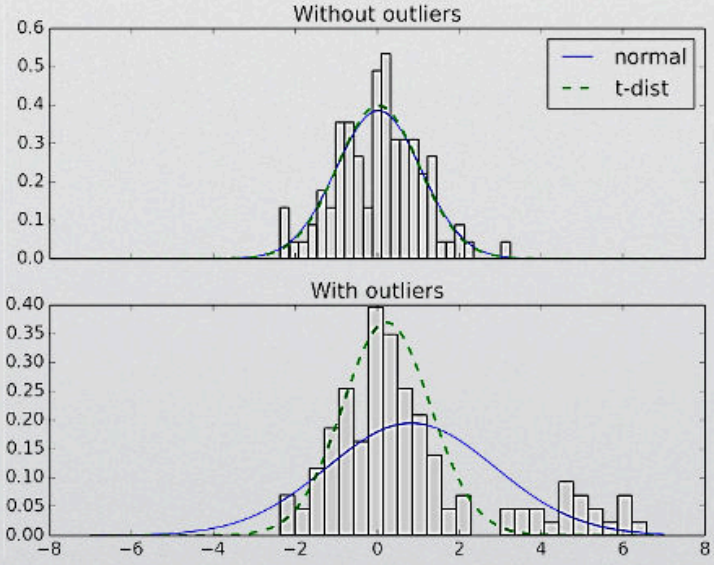


对于拥挤问题(The Crowding Problem)的解决，提出t-SNE，一种非线性降维策略，主要用于可视化数据。引入厚尾部的学生t分布，将低维空间映射点之间的距离度量转化为概率分布t分布qij，使得不同簇之间的点能很好地分开。

1.2 t-SNE

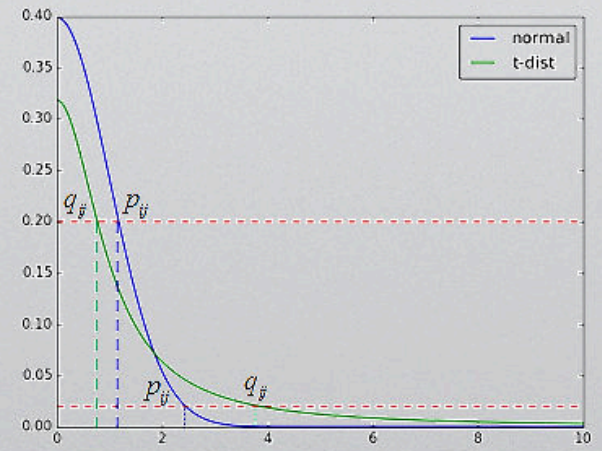
➢ t-SNE

从前两张图中可以看到，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，因此其拟合结果更为合理，较好的捕获了数据的整体特征。



图中有高斯分布和t分布两条曲线，表示点之间的相似性与距离的关系，高斯分布对应高维空间，t分布对应低维空间。那么对于高维空间中相距较近的点，为了满足$p_{ij}=q_{ij}$，低维空间中的距离需要稍小一点；而对于高维空间中相距较远的点，为了满足$p_{ij}=q_{ij}$，低维空间中的距离需要更远。这恰好满足了我们的需求，即同一簇内的点(距离较近)聚合的更紧密，不同簇之间的点(距离较远)更加疏远。我们使用自由度为1的t分布重新定义q_{ij}：

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



1.3 Deep Embedded Clustering(DEC)

受t-SNE的启发，提出DEC算法，重新定义pij，它是根据qij得到的，相当于对qij增加权重，使得数据更尖锐化，隐层软分配凸的更凸。微调阶段，舍弃掉编码器层，最小化KL散度作为损失函数，迭代更新参数。DEC通过降噪自编码，逐层贪婪训练后组合成栈式自编码，然后撤去解码层，仅使用编码层，对提取出来的特征使用相对熵作为损失函数对网络进行微调，该结构可以同时对数据进行特征学习和聚类。但是DEC算法没有考虑微调会扭曲嵌入式空间，削弱嵌入式特征的代表性，从而影响聚类效果。

DEC算法先使用整个网络进行预训练，得到原始数据经过非线性映射到潜在特征空间的数据表示，即特征。然后对得到的特征用K-means算法进行网络初始化,得到初始聚类中心。再使用相对熵迭代，微调网络，直至满足收敛性判定准则停止。

补充一点，在得到隐层特征z之后，外加一层聚类层，聚类中心μ就是z与聚类层的连接权重。通过聚类层，得到KL散度损失函数。

➢ Deep Embedded Clustering(DEC)

q_{ij} is the similarity between embedded point z_i and cluster center μ_j measured by Student's t-distribution.

$$q_{ij} = \frac{(1+\|z_i-\mu_j\|^2)^{-1}}{\sum_j (1+\|z_i-\mu_j\|^2)^{-1}}$$

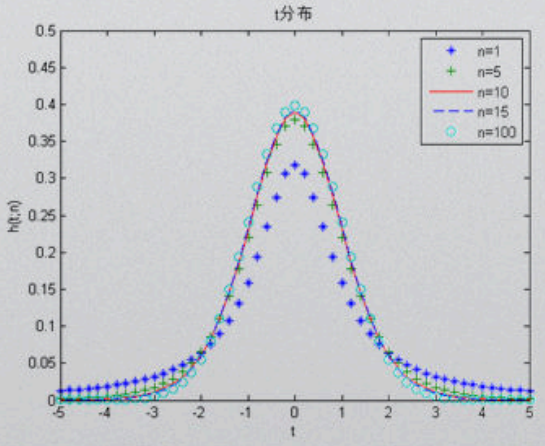
$$p_{ij} = \frac{q_{ij}^2 / \sum_j q_{ij}}{\sum_j (q_{ij}^2 / \sum_j q_{ij})}$$

计算p_i首先将q_i提升到2次方，然后通过每个簇的频率进行归一化。

（1） 加强预测（即提高簇纯度）。q分布为软分配的概率，那么p如果使用delta分布来表示，显得比较原始。

（2） 更加重视高置信度的数据点。置信度越高，属于某个聚类概率越大。

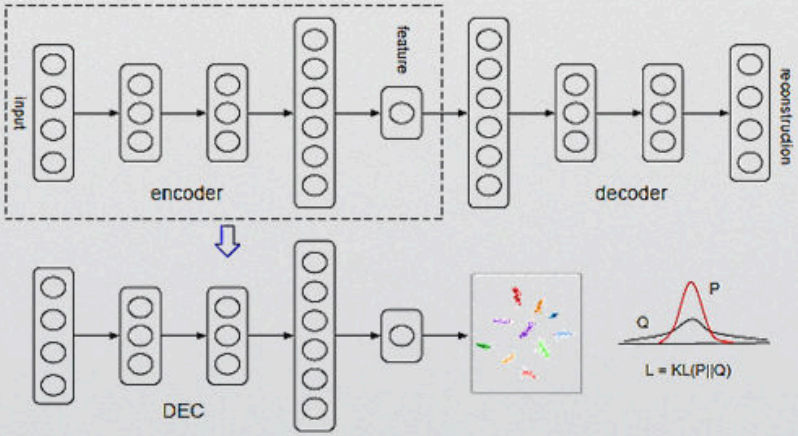
（3） 规范化每个质心的损失贡献以防止大簇扭曲隐藏特征空间。



$$L = \sum_i^N \sum_j^k p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

$$\frac{\partial L}{\partial z_i} = \frac{\alpha+1}{\alpha} \sum_j (1+\frac{\|z_i-\mu_j\|^2}{\alpha})^{-1} \times (p_{ij}-q_{ij})(z_i-\mu_j),$$

$$\frac{\partial L}{\partial \mu_j} = -\frac{\alpha+1}{\alpha} \sum_i (1+\frac{\|z_i-\mu_j\|^2}{\alpha})^{-1} \times (p_{ij}-q_{ij})(z_i-\mu_j).$$



$$f_\theta : X \rightarrow Z.$$

经过贪婪的逐层训练后，我们以反向逐层训练顺序将所有编码器层与所有解码器层连接起来，形成一个深层自动编码器，然后对其进行微调，以最大程度地减少重构损失。最终结果是深层自动编码器，中间有一个瓶颈编码层。然后，我们丢弃解码器层，并将编码器层用作数据空间和特征空间之间的初始映射。

剩下的编码器层通过优化L交替迭代进行微调：

Step 1: 计算嵌入点与聚类中心间的软分配q_{ij} (当某点与某聚类中心依概率符合分布时,将其分配给该中心);

Step 2: 更新映射f，通过从"使用辅助目标分布的高置信分配"中学习来细化聚类中心μ.

这个过程直到某种收敛准则符合而停止。

Xie J, Girshick R, Farhadi A. Unsupervised deep embedding for clustering analysis[C]//International conference on machine learning. 2016: 478-487.

2. Improved Deep Embedded Clustering(IDEC)

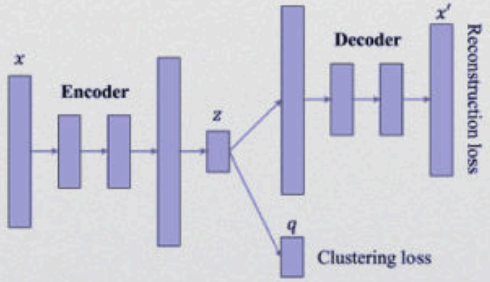
DEC丢弃解码器层，并使用聚类损失Lc对编码器进行微调。作者认为这种微调会扭曲嵌入空间，削弱嵌入特征的代表性，从而影响聚类性能。因此，提出保持解码器层不变，直接将聚类损失附加到嵌入空间。IDEC算法是对DEC算法的改进，通过保存局部结构防止微调对嵌入式空间的扭曲，即在预训练时，使用欠完备自编码，微调时的损失函数采用相对熵和重建损失之和，以此来保障嵌入式空间特征的代表性。

基于局部结构保留的深度嵌入聚类IDEC是对DEC算法的改进，通过保存局部结构方式避免微调时对嵌入空间的扭曲。IDEC算法在预训练结束后，对重建损失和聚类损失的加权和进行微调，在最大限度保证不扭曲嵌入空间的前提下，得到最优聚类结果。

➢ Improved Deep Embedded Clustering(IDEC)

DEC的工作原理是使用高置信度的样本作为监督，然后使每个簇中的样本分布更密集。然而，并不能保证将边缘点正确聚类。本文通过显式地保留数据的局部结构来处理这个问题。这样，高置信度样本的监督信息可以帮助边缘样本正确聚类。

自编码得到的嵌入点不一定适合聚类任务。为此，DEC丢弃解码器层，并使用聚类损失Lc对编码器进行微调。但是，我们认为这种微调会扭曲嵌入空间，削弱嵌入特征的代表性，从而影响聚类性能。因此，我们建议保持解码器层不变，直接将聚类损失附加到嵌入空间。



$$L = \| \mathbf{H}^{(M)} - \mathbf{X} \|_F^2 + \gamma \sum_i^N \sum_j^k p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Follow suggestions in [Xie *et al.*, 2016], we also pretrain a stacked denoising autoencoder before performing clustering. After pretraining, embedded points are valid feature representations for input samples. Then cluster centers $\{\mu_j\}_{j=1}^K$ can be initialized by employing k-means on $\{z_i = f_W(x_i)\}_{i=1}^n$

$$q_{ij} = \frac{(1+\|z_i - \mu_j\|^2)^{-1}}{\sum_j (1+\|z_i - \mu_j\|^2)^{-1}} \qquad p_{ij} = \frac{q_{ij}^2 / \sum_j q_{ij}}{\sum_j (q_{ij}^2 / \sum_j q_{ij})}$$

$$s_i = \arg\max_j q_{ij}$$

Input: Input data: X; Number of clusters: K; Target distribution update interval: T; Stopping threshold: δ; Maximum iterations: $MaxIter$.
Output: Autoencoder's weights W and W'; Cluster centers μ and labels s.

1 Initialize μ, W' and W according to Section 3.1.
2 **for** $iter \in \{0, 1, \ldots, MaxIter\}$ **do**
3 **if** $iter \% T == 0$ **then**
4 Compute all embedded points $\{z_i = f_W(x_i)\}_{i=1}^n$
5 Update P using (3), (4) and $\{z_i\}_{i=1}^n$.
6 Save last label assignment: $s_{old} = s$.
7 Compute new label assignments s via (14).
8 **if** $sum(s_{old} \neq s)/n < \delta$ **then**
9 Stop training.
10 Choose a batch of samples $S \in X$.
11 Update μ, W' and W via (11), (12) and (13) on S.

Guo X, Gao L, Liu X, et al. Improved deep embedded clustering with local structure preservation[C]//IJCAI. 2017: 1753-1759.

代码中q计算了两遍，进batch之前计算了所有的q，并根据q得到总体的p。在每个batch里面，又更新了各自的q，但是p仍然调用之前的p(从之前的p中找到该batch相应序号的p)，这是由于batch里面无法获取全部的更新的q的信息，也就无法计算p。在每个batch中计算每个batch的目标函数，p没有根据q更新而更新，不过下一个epoch会更新p。

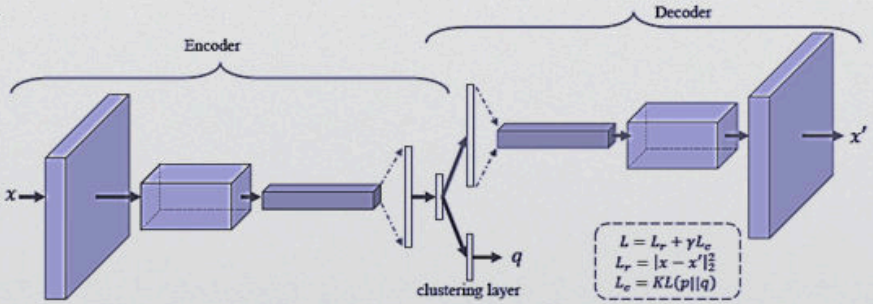
3. Deep Convolutional Embedded Clustering(DCEC)

深度卷积嵌入聚类算法(deep convolutional embedded clustering, DCEC),是在DEC原有网络基础上，加入了卷积自编码操作，并在特征空间保留数据局部结构，从而取得了更好聚类效果。

深度卷积嵌入聚类算法DCEC是在IDEC算法基础上进行的改进，将编码层和解码层中的全连接换成卷积操作，这样可以更好地提取层级特征。图中编码层和解码层各有3层卷积，卷积层后加了一个flatten操作拉平特征向量，以获得10维特征。DCEC只是将IDEC的所有全连接操作换成卷积操作，其损失函数依旧是重建损失和聚类损失之和。

➢ *Deep Convolutional Embedded Clustering(DCEC)*

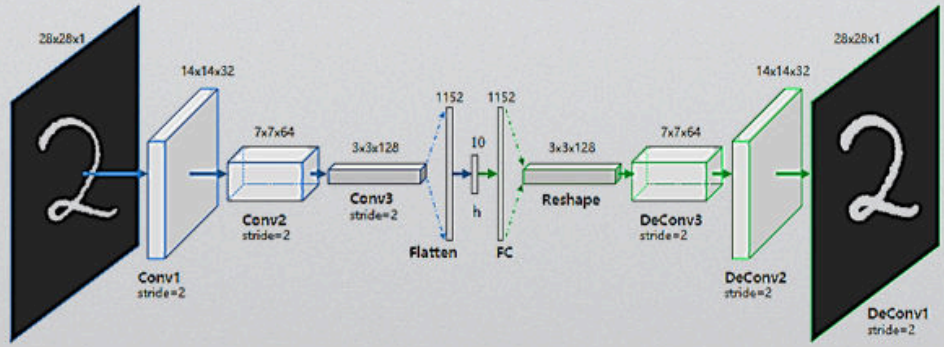
深度卷积嵌入聚类算法DCEC是在IDEC算法基础上进行的改进,将编码层和解码层中的全连接换成卷积操作,这样可以更好地提取层级特征。图中编码层和解码层各有3层卷积,卷积层后加了一个flatten操作拉平特征向量,以获得10维特征。DCEC只是将IDEC的所有全连接操作换成卷积操作,其损失函数依旧是重建损失和聚类损失之和。



$$L = \| \mathbf{H}^{(M)} - \mathbf{X} \|_F^2 + \gamma \sum_i^N \sum_j^k p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2)^{-1}}{\sum_j (1 + \|z_i - \mu_j\|^2)^{-1}} \qquad p_{ij} = \frac{q_{ij}^2 / \sum_j q_{ij}}{\sum_j (q_{ij}^2 / \sum_j q_{ij})}$$

$$s_i = \arg\max_j q_{ij}$$

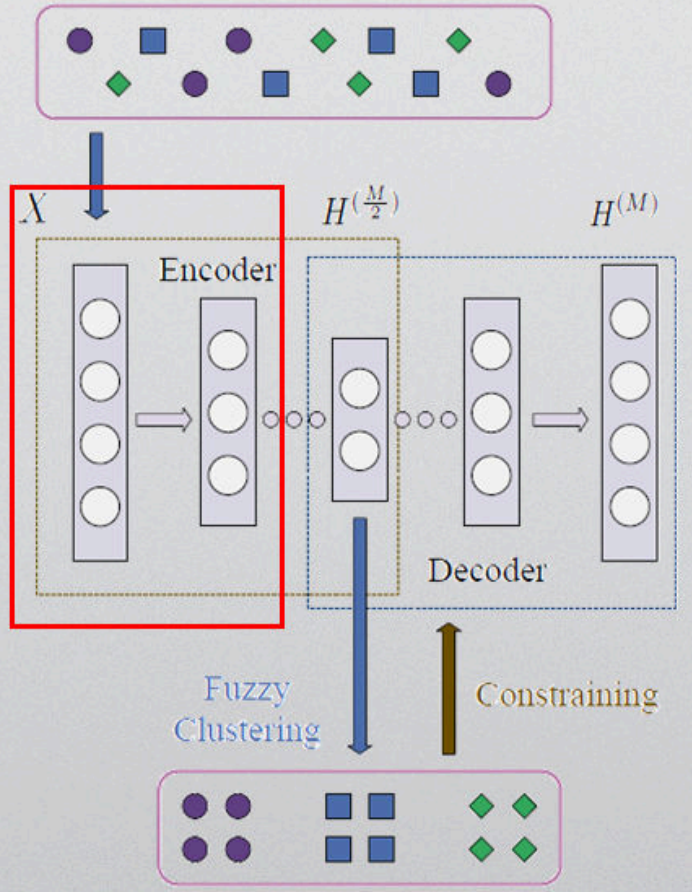


Guo X, Liu X, Zhu E, et al. Deep clustering with convolutional autoencoders[C]//International Conference on Neural Information Processing. Springer, Cham, 2017: 373-382.

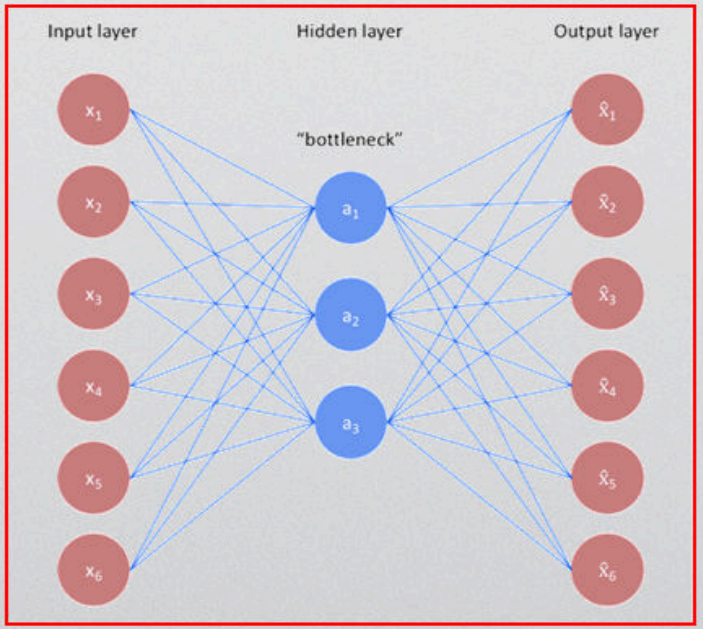
4. Deep Fuzzy K-means(DFKM)

Deep Fuzzy K-means同样在低维映射空间中加入聚类过程，将特征提取与聚类同时进行，引入熵加权的模糊K-means，不采用原来的欧氏距离，而是自己重新定义度量准则，权值偏置的正则化项防止过拟合，提高泛化能力。

➢ *Deep Fuzzy K-means*



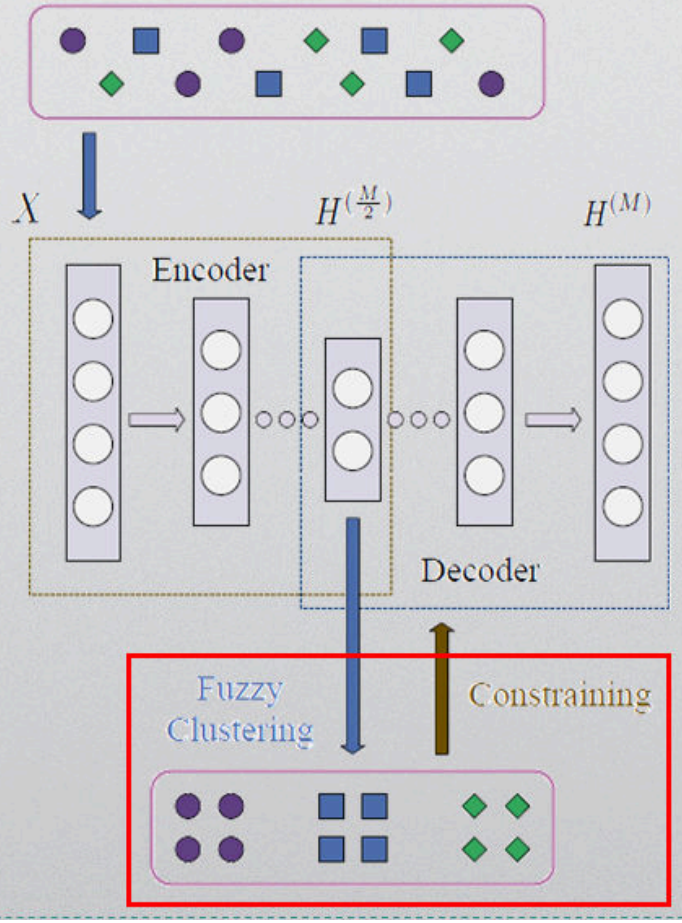
(1) 逐层预训练（自编码）



(2) 精调

$$\min_{\mathbf{W}^{(m)},\mathbf{b}^{(m)}} \mathcal{F} = \| \mathbf{H}^{(M)} - \mathbf{X} \|_F^2$$

➢ Deep Fuzzy K-means



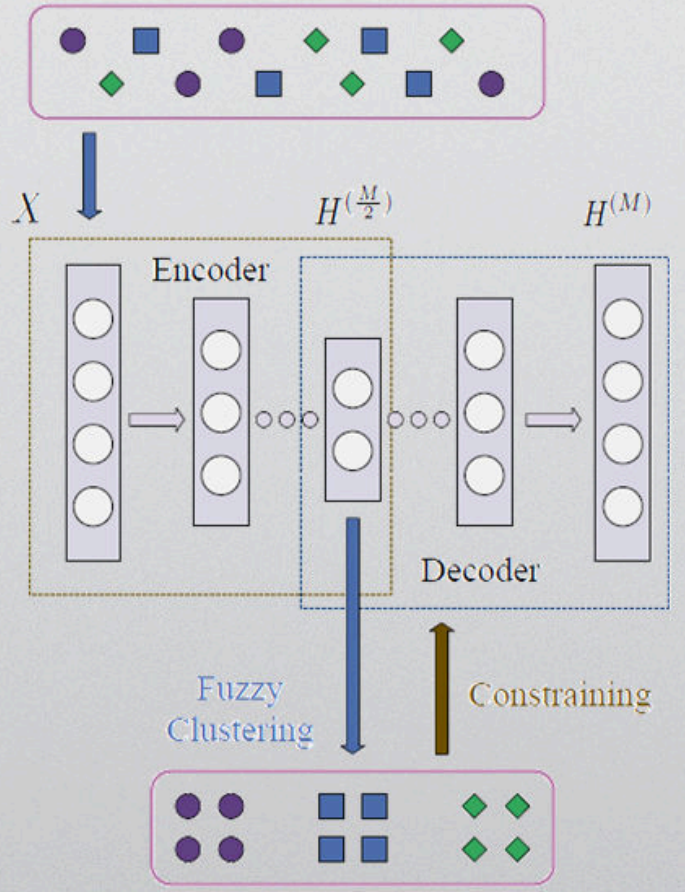
✓ Fuzzy K-Means

$$\min_{\mathbf{U},\mathbf{C}} \sum_{i=1}^{N}\sum_{j=1}^{k} u_{ij} \| \mathbf{h}_i^{(\frac{M}{2})} - \mathbf{c}_j \|_{\hat{\sigma}} + \gamma u_{ij} \log u_{ij}$$

$$s.t. \quad \sum_{j=1}^{k} u_{ij} = 1, \; 0 < u_{ij} < 1$$

$$\| g(x) \|_{\hat{\sigma}} = \frac{(1+\sigma)\| g(x) \|_2^2}{\| g(x) \|_2 + \sigma}$$

➢ Deep Fuzzy K-means



Loss function

$$\min_{\mathbf{W}^{(m)},\mathbf{b}^{(m)},\mathbf{U},\mathbf{C}} \| \mathbf{H}^{(M)} - \mathbf{X} \|_F^2$$

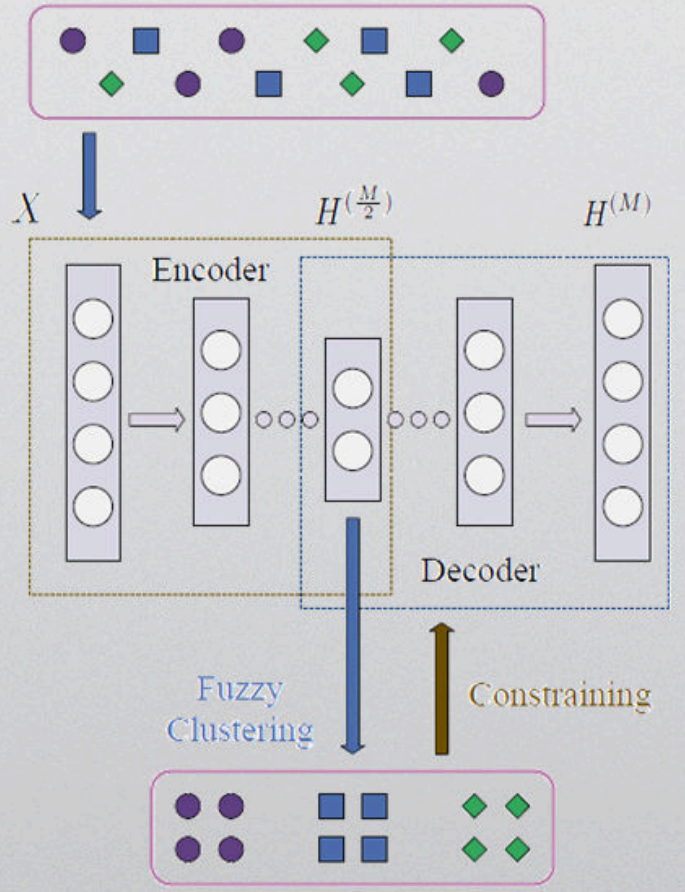
$$+ \lambda_1 \sum_{i=1}^{N} \sum_{j=1}^{k} u_{ij} \| \mathbf{h}_i^{(\frac{M}{2})} - \mathbf{c}_j \|_{\hat{\sigma}} + \gamma u_{ij} \log u_{ij}$$

$$+ \lambda_2 \sum_{m=1}^{M} \| \mathbf{W}^{(m)} \|_F^2 + \| \mathbf{b}^{(m)} \|_2^2$$

$$s.t. \quad \sum_{j=1}^{k} u_{ij} = 1, \; 0 < u_{ij} < 1$$

where λ_1 and λ_2 are tradeoff parameters. $\mathbf{c}_j \in \mathbb{R}^{d'}$ is the j-th cluster centroid in low-dimensional feature space with $d' = d^{(\frac{M}{2})}$.

Deep Fuzzy K-means



Input: Input data matrix $\mathbf{X}$, the assignment matrix $\mathbf{U}_0$ obtained from fuzzy k-means or other approaches, the number of clusters k, parameters γ, μ, σ, λ_1 and λ_2, SGD maximum iterations L.

Initialize: $\mathbf{H}^{(0)} = \mathbf{X}$, $\mathbf{U} = \mathbf{U}_0$, that satisfies $\sum_{j=1}^{k} u_{ij} = 1$, a random matrix $\mathbf{C} \in \mathbb{R}^{d' \times k}$ where $d' = d^{(\frac{M}{2})}$, random matrices $\mathbf{W}^{(m)}$ and random vectors $\mathbf{b}^{(m)}$ where $m = 1, 2, \cdots, M$.

Pre-train the auto-encoder.

repeat

Calculate d_{ij} by Eq.(23) for $i = 1, 2, \cdots, N$ and $j = 1, 2, \cdots, k$.

for $i = 1, 2, \cdots, L_1$ **do**

Update $\mathbf{W}^{(m)}$ and $\mathbf{b}^{(m)}$ by Eq.(27) for $m = 1, 2, \cdots, M$.

end for

Update $\mathbf{C}$ by (28).

Update $\mathbf{U}$ by (31).

until $\mathbf{C}$, $\mathbf{U}$ converge or exceed the maximum iterations

Output: clustering assignment matrix $\mathbf{U}$, centroid matrix $\mathbf{C}$ and parametric neural network with $\mathbf{W}^{(m)}$ and $\mathbf{b}^{(m)}$.

5. 参考文献

[1] Maaten L, Hinton G. Visualizing data using t-SNE[J]. Journal of machine learning research, 2008, 9(Nov): 2579-2605.

[2] Vincent P, Larochelle H, Lajoie I, et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion[J]. Journal of machine learning research, 2010, 11(Dec): 3371-3408.

[3] Xie J, Girshick R, Farhadi A. Unsupervised deep embedding for clustering analysis[C]//International conference on machine learning. 2016: 478-487.

[4] Guo X, Gao L, Liu X, et al. Improved deep embedded clustering with local structure preservation[C]//IJCAI. 2017: 1753-1759.

[5] Guo X, Liu X, Zhu E, et al. Deep clustering with convolutional autoencoders[C]//International Conference on Neural Information Processing. Springer, Cham, 2017: 373-382.

[6] Zhang R, Li X, Zhang H, et al. Deep Fuzzy K-Means with Adaptive Loss and Entropy Regularization[J]. IEEE Transactions on Fuzzy Systems, 2019.

[7] t-SNE相关资料：t-SNE完整笔记、An illustrated introduction to the t-SNE algorithm、从SNE到t-SNE再到LargeVis、t-SNE算法-CSDN

[8] DEC与IDEC的Python代码-Github、DEC-Keras-Github、piiswrong/dec-Github、DCEC-Github

[9] DFKM的Python代码-Github

[10] 谢娟英，侯琦，曹嘉文. 深度卷积自编码图像聚类算法[J]. 计算机科学与探索, 2019, 13(4): 586-595.DOI:10.3778/j.issn.1673-9418.1806029.

[11] Deep Clustering: methods and implements-Github 深度聚类会议论文汇总

[12] Deep Clustering | Deep Learning Notes