Generalized Autoencoder: A Neural Network Framework for Dimensionality Reduction

Wei Wang, Yang Huang, Yizhou Wang, Liang Wang

Part 1
Dimensionality Reduction Algorithms

Dimensionality Reduction

Given a dataset $X = \{x_1, x_2, \dots, x_N\}$, $x_i \in \mathbb{R}^D$, where N is the number of samples and D is the number of features.

Find a mapping function $F: x \longrightarrow y$ that transforms $x \in \mathbb{R}^D$ into the desired low-dimensional representation $y \in \mathbb{R}^d$.

The motivation **: to recover intrinsic dimensionality of the data**.

The **intrinsic dimension** for a data set can be thought of as the number of variables needed in a minimal representation of the data.

Graph Embedding and Extensions: A General Framework for Dimensionality Reduction
https://en.wikipedia.org/wiki/Intrinsic_dimension

Dimensionality Reduction Algorithms

Linear dimensionality reduction, known as “subspace learning” :
Subspace learning assumes intrinsic dimensionality can be obtained by linear projection

- Principle Component Analysis (PCA) 主成分分析
- Linear Discriminant Analysis (LDA) 线性判别分析
- Multidimensional Scaling (MDS) 多维尺度变换

However,
Linear methods will fail if the data lies on a low-dimensional manifold because the data structure becomes highly nonlinear.

Nonlinear dimensionality reduction (NLDR), known as “ Manifold learning”
Manifold learning can be defined as a process that automatically learns the geometric and topological properties of a given manifold.

- Isometric Feature Mapping (ISOMAP) 等距映射
- Locally Linear Embedding (LLE) 局部线性嵌入
- Laplacian Eigenmaps (LE) 拉普拉斯映射
- …

Part 1.1
Subspace Learning

Subspace Learning - PCA

PCA is based on computing the low-dimensional representation that maximize data variance.

Find the linear transformation matrix $\widehat{W} \in \mathbb{R}^{D \times d}$

$$\widehat{W} = \underset{W^T W = I}{\operatorname{argmax}} Tr(W^T Cov(X) W)$$

PCA可看成是一个坐标变换的过程，将高维数据的坐标投影到数据方差最大的方向组成的新坐标中。

https://www.cnblogs.com/mikewolf2002/articles/3429711.html

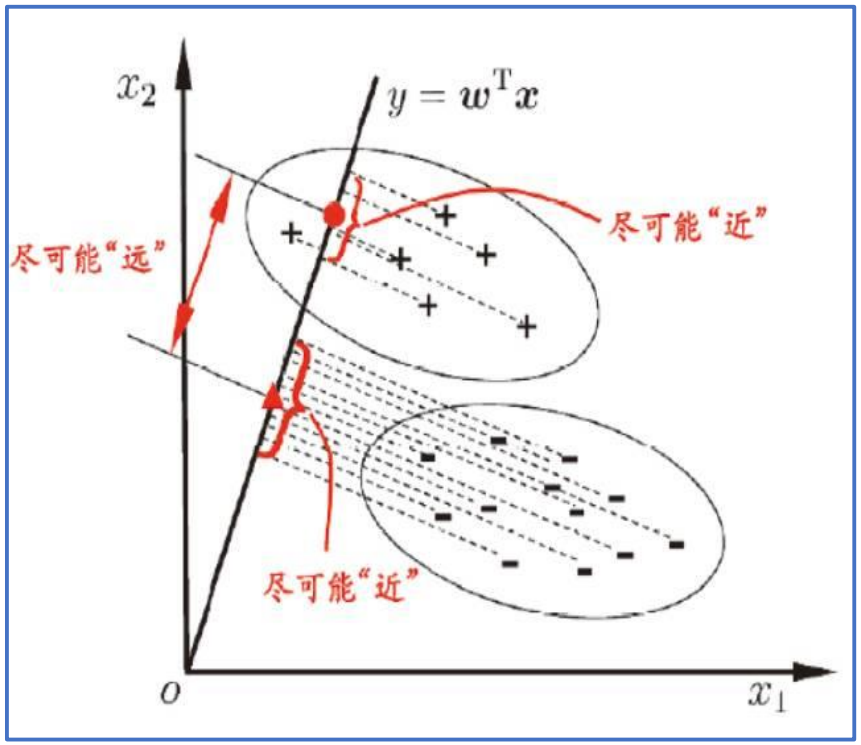
Subspace Learning - LDA

线性判决分析 (LDA) 在降维过程中考虑了数据的类别信息，目标是寻找能对数据进行有效分类的方向。
思想：投影后类内的方差最小，类间方差最大。
方法：构造两个矩阵，类内散度矩阵 S_W 和类间散度矩阵 S_B

$$S_W = \sum_{i=1}^{c}\sum_{j}^{n_i}(x_j^i - m^i)(x_j^i - m^i)^T$$

$$S_B = \sum_{i=1}^{c} n_i(m^i - m)(m^i - m)^T$$

n_i表示第i类样本的个数，x_j^i表示第i类样本中的第j个样本。
m^i表示第i 类样本的平均值，m表示总体样本均值，c为类别数。



https://blog.csdn.net/ruthywei/article/details/83045288

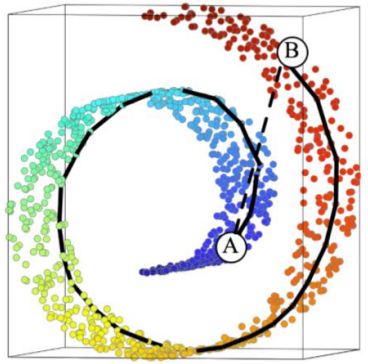
LDA的判决准则定义为：

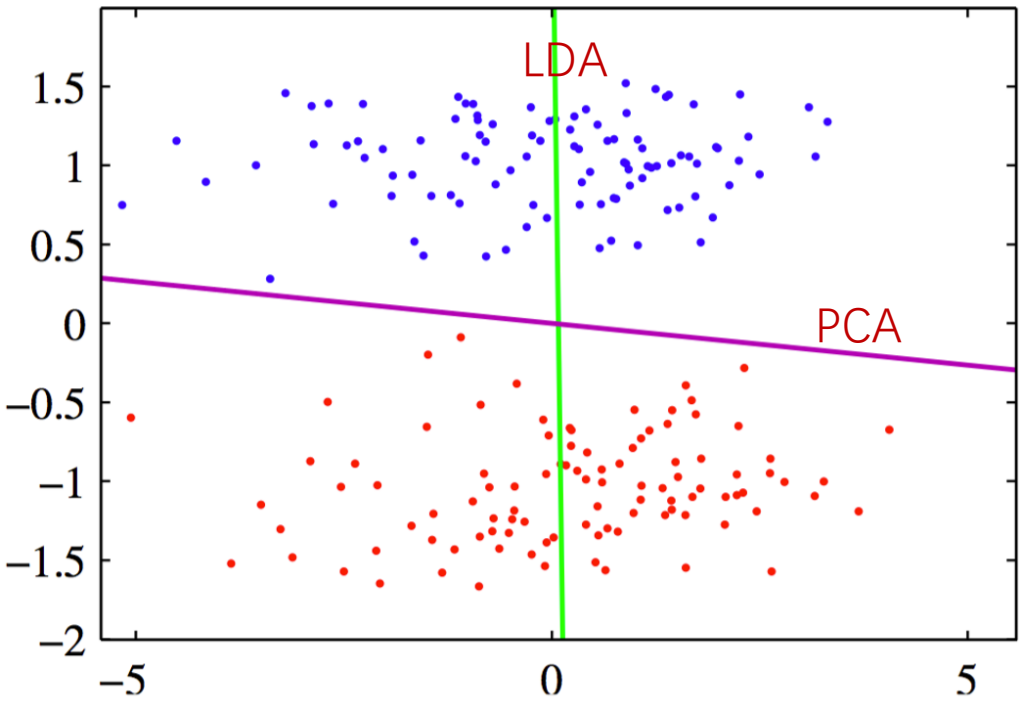
$$\underset{W_{LDA}}{\operatorname{argmax}} \frac{Tr(W_{LDA}^T S_B W_{LDA})}{Tr(W_{LDA}^T S_w W_{LDA})}$$

$$s.t. W_{LDA}^T W_{LDA} = I_d$$

这里的W_{LDA}表示投影矩阵。

PCA vs LDA





图片来自Graph Embedding and Extensions: A General Framework for Dimensionality Reduction

Advantages:

- Both PCA and LDA are non-parametric, meaning they require no free parameters to set.
- They are computationally more efficient when data lies on a linear subspace and $D < N$

Disadvantages:

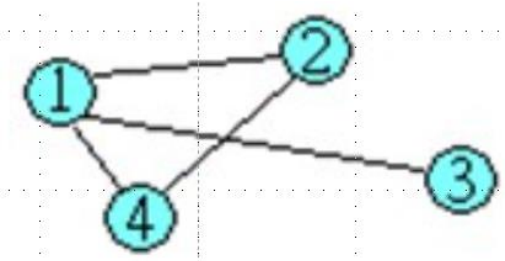
- They are linear models and fail when the true underlying structure of the data is on a nonlinear manifold. Such as “Swiss Roll”
- They assume data distributions are Gaussian, making the algorithm sensitive to outliers.
- LDA requires the number of available projection directions is lower than the class number.

Graph Embedding

Given a dataset $X = \{x_1, x_2, \ldots, x_N\}$,
Let G be an undirected weighted graph (two-way direction) with vertex set X (N nodes) and similarity (or weighted) matrix $W \in R^{N \times N}$

W :

- Symmetric matrix
- May be negative



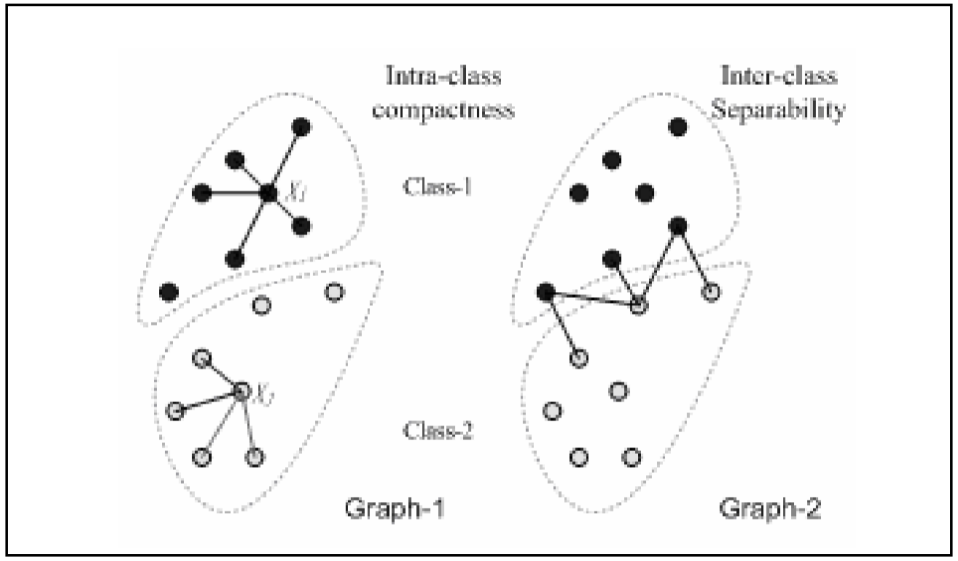
$$X = [x_1 \quad x_2 \quad x_3 \quad x_4]$$
$$W \in R^{4 \times 4}$$

$$W = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \qquad D = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

Subspace Learning - MFA

- Marginal Fisher Analysis (MFA) extends LDA
- LDA assumes data distribution of each class to be Gaussian.
- MFA aims to remove this strong assumption by developing new criteria that characterize intraclass compactness and interclass separability.

类内紧致性（Intraclass compactness） 表示为同一类的每个点与其k-最近邻点之间的距离之和。
类间可分离性（Interclass separability） 表示不同类的边缘点与其邻接点之间的距离之和。



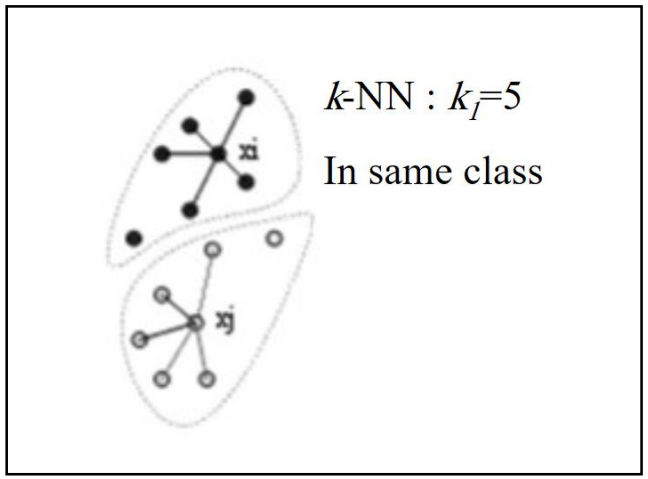
Graph embedding: a general framework for dimensionality reduction （2005）

Subspace Learning - MFA

Intra-class compactness 类内紧致性

$$\tilde{S}_c = \sum_i \sum_{i\in N_{k_1}^+(j) or\ j\in N_{k_1}^+(i)} \| w^{\mathrm{T}}x_i - w^{\mathrm{T}}x_j \|^2$$
$$= 2w^{\mathrm{T}}X(D^c - W^c)X^{\mathrm{T}}w \qquad (8)$$
$$W_{ij}^c = 1 \quad if\ j\in N_{k_1}^+(i)\ or\ i\in N_{k_1}^+(j);\ 0, else.$$

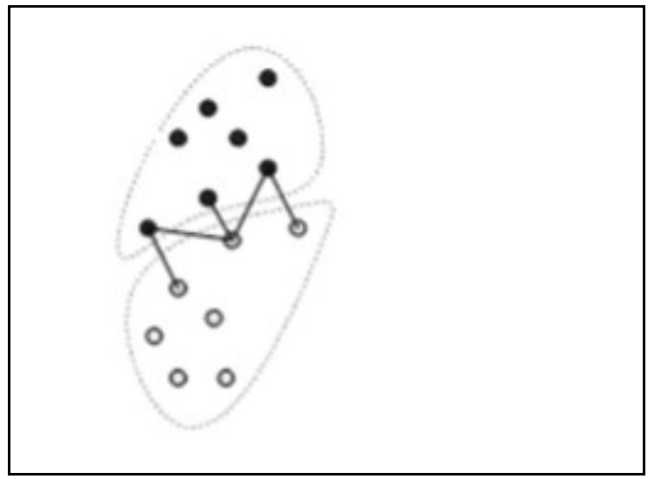
where $N_{k_1}^+(i)$ means the k_1 nearest neighbors in the same class of the sample x_i .



Inter-class separability 类间可分离性

$$\tilde{S}_m = \sum_i \sum_{(i,j)\in P_{k_2}(l_i) or (j,i)\in P_{k_2}(l_j)} \| w^{\mathrm{T}}x_i - w^{\mathrm{T}}x_j \|^2$$
$$= 2w^{\mathrm{T}}X(D^m - W^m)X^{\mathrm{T}}w$$
$$W_{ij}^m = 1\ \ if\ (i,j)\in P_{k_2}(l_i)\ or\ (j,i)\in P_{k_2}(l_j);\ \ 0, else.$$

where $P_{k_2}(c)$ is a set of data pairs that are the k_2 nearest pairs among $\{(i,j), i\in \pi_c, j\notin \pi_c\}$.

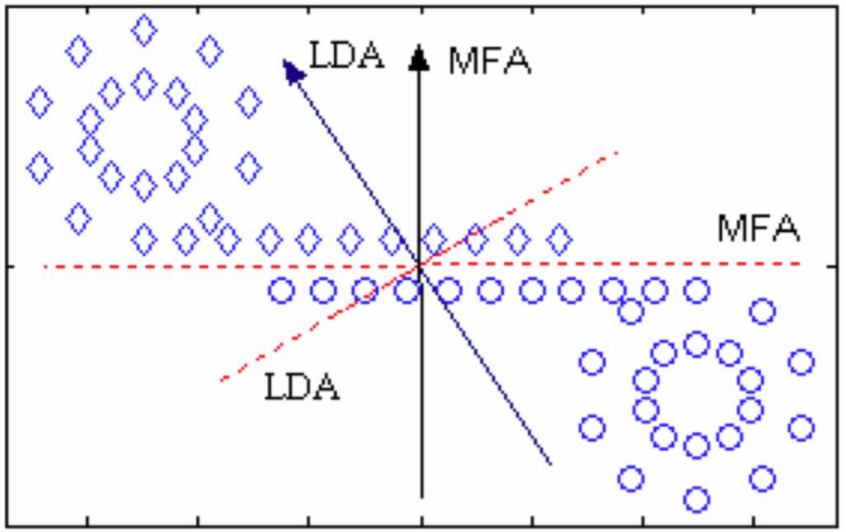


Graph embedding: a general framework for dimensionality reduction （2005）

Subspace Learning - MFA

Cost function:

$$w^* = arg\min_w \frac{\tilde{S}_c}{\tilde{S}_p} = arg\min_w \frac{w^T X(D^c - W^c)X^T w}{w^T X(D^m - W^m)X^T w}$$



Graph embedding: a general framework for dimensionality reduction (2005)

Subspace Learning - MDS

MDS 使原始样本空间中样本之间的距离在低维空间得以保持。
假设高维数据集X中的样本间的不相似矩阵为$\sum = (\sigma_{ij})_{n\times n}$, MDS的目标使寻求低维数据表示$Y = \{y_i, i = 1,2,\dots,n\}$使得低维数据点对之间的距离$d_{ij}$尽可能接近$\sigma_{ij}$.
若采用的度量是欧式度量，则有

$$\sigma_{ij}{}^2 = d_{ij}{}^2 = \|x_i - x_j\|^2 = x_i^T X_i - 2\, x_i^T X_j + x_j^T X_j$$

令$P = (x_1^T X_1, \dots, x_n^T X_n)^T$, $e = (1,1,\dots,1)^T$，则

$$D = Pe^T - 2X^T X + eP^T$$

其中，D为距离矩阵。再令$H = I - ee^T/N$为数据中心化矩阵，则有

$$N = -\frac{HPH}{2} = X^T X$$

记N的特征值分解为

$$N = USU^T$$

其中，U为正交矩阵，S为特征值按降序排列的对角矩阵，最终的低维表示为

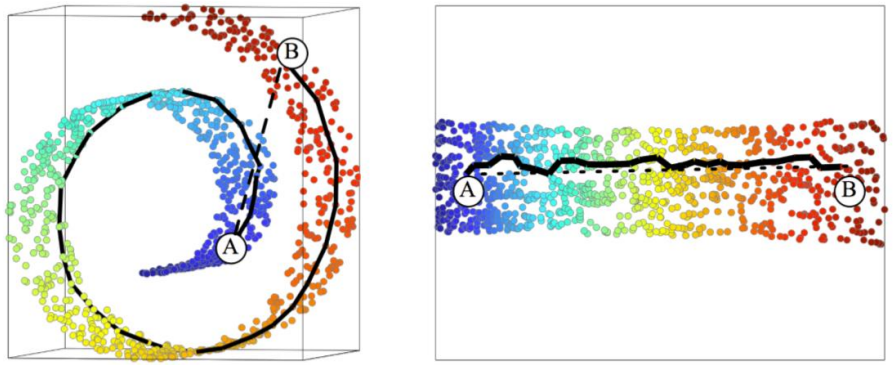
$$Y = S^{1/2} U_d$$

其中，U_d表示U中前d个最大非零特征值所对应特征向量组成的矩阵。

Part 1.2
Manifold Learning

Manifold Learning - ISOMAP

等距映射（ISOMAP）是以多维尺度空间（MDS）为基础，其特点是保持空间上两点间的测地线距离（Geodesic Distance).主要思想是通过Dijkstra算法构建邻域图，然后用图中的最短路径即测地线来代替欧式距离，从而在低维空间对流形进行重构。



ISOMAP 的主要计算步骤如下[1]：

（一） 构建高维空间中所有数据点的邻接图 G，距离定义为欧式距离，邻接关系定义为 ε 或 K 邻域，为了方便计算所以一般选择使用 K 邻域。

（二） 通过快速算法计算图 G 上两点间的最短路径 $d_G(x_i,x_j)$，用 $d_G(x_i,x_j)$ 来表示流形上两两点间的测地线距离，从而得到图 G 上任意两点间的最短路径距离矩阵 $D_G=\{d_G(x_i,x_j)\}$。

（三） 使用 MDS 算法，用测地线距离矩阵 D_G 在 d 维欧式空间中对流形进行重构。令 $S=(S_{ij})=(D_{ij}^2)$，$H=(H_{ij})=(\delta_{ij}-1/N)$，$\tau(D)=-HSH/2$，$\tau(D)$ 的最大 d 个特征值 $\lambda_1,\lambda_2,...,\lambda_d$ 及其对应的特征向量 $v_1,v_2,...,v_d$ 所组成的矩阵 $V=[v_1,v_2,...,v_d]$，那么流形在 d 维欧式空间中的嵌入结果为 $Y=diag(\sqrt{\lambda_1},\sqrt{\lambda_2},...,\sqrt{\lambda_d})V^T$。

Manifold Learning - LLE

局部线性嵌入(LLE)主要考虑保持高维数据集的局部结构信息，即局部线性关系。并且希望这种关系在低维空间中得以保持。

主要过程:

- 对每个高维数据点$x_i(i = 1,2,\dots,n)$,寻找k个最近邻，通过建立邻域图来描述原始数据的集合关系。

$$x_i = w_{ij}x_j + w_{ik}x_k + w_{il}x_l$$

- 计算能够最佳重构样本点$x_i(i = 1,2,\dots,n)$的权值W_{ij}，即最小化重构误差:

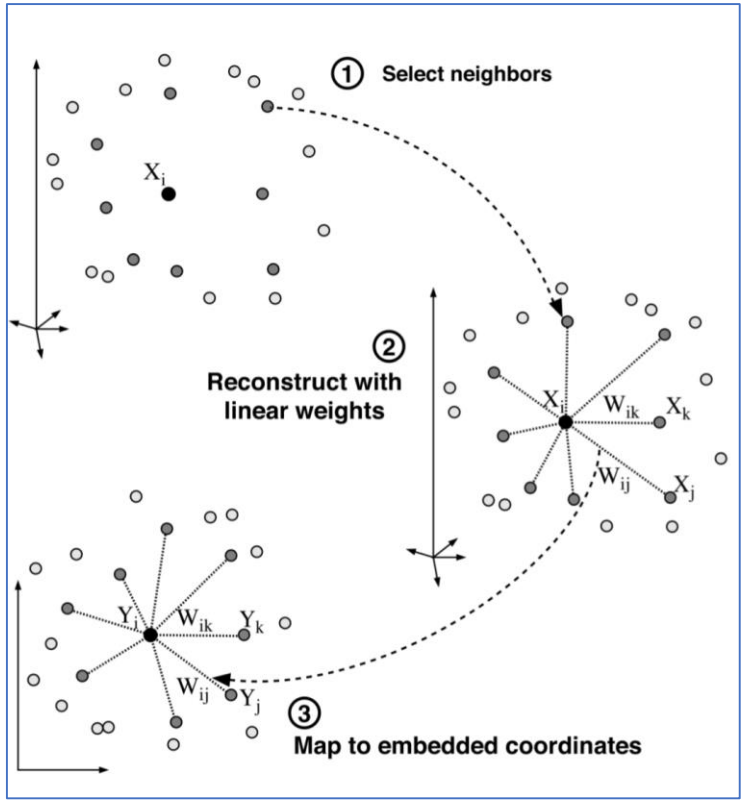
$$\underset{W}{\operatorname{argmin}} \left\| x_i - \sum_j W_{ij}x_j \right\|^2$$

其中，对权值W_{ij}有两个约束:

(1) 当x_j不是x_i的近邻数据点时$W_{ij} = 0$;　　(2) $\sum_j W_{ij} = 1$.

- 依据假设，低维数据点y_i仍然可以通过其对应的k个最近邻点重构，目标函数为:

$$\underset{y_i}{\operatorname{argmin}} \|\varepsilon(y)\|^2 = \underset{y_i}{\operatorname{argmin}} \left\| y_i - \sum_j W_{ij}y_j \right\|^2$$



《机器学习》 周志华

Manifold Learning - LE

拉普拉斯特征映射 (LE) 核心思想是通过图嵌入算法来保持数据点之间的局部结构。
直观地说，LE期望在高维空间中离得很近的数据点对应于低维空间中的投影点的距离也应该离得很近。

算法步骤：

- 计算高维空间中给定点x_i的k个近邻点，构造一个无向邻域图G. G的节点表示n个样本点，它的所对应边表示数据点之间的近邻关系。
- 计算近邻数据点的权值，构建出权值矩阵W_{ij}。权值矩阵的构造一般采取两种方式：

  (1) 若数据点x_i和x_j在邻域图G中互为近邻点，则W_{ij}=1，否则W_{ij}=0

  (2) 热核函数：x_i和x_j在邻域图G中互为近邻点，则$W_{ij} = exp\frac{-\|x_i - x_j\|^2}{\sigma}$，否则$W_{ij}$=0
- 计算低维表示。LE目标函数定义为：

$$\phi(Y) = min \sum_{i,j} (y_i - y_j)^2 W_{ij}$$

其中 $\sum_{i,j}(y_i - y_j)^2 W_{ij} = Y^T L Y$,

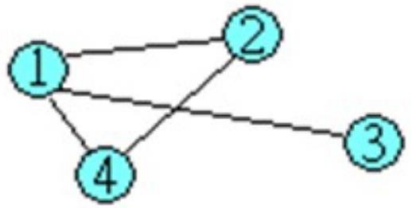
$L = D - W$被称为拉普拉斯算子，其中D为对角矩阵$D_{ij} = \sum_j W_{ij}$

LE-拉普拉斯算子

L=Degree-Adjacent=>

$$L = D - W$$
$$D_{ii} = \sum_{i \neq j} W_{ij}, \forall i. \quad (2)$$

G={X,W} :



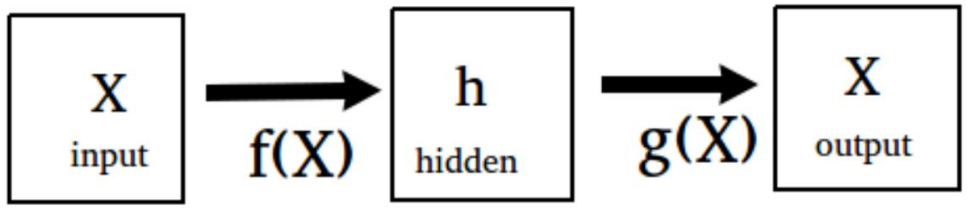
$$W = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \quad L = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ -1 & -1 & 0 & 2 \end{bmatrix}$$

W is weighted matrix ,also call similarity matrix

Part 2
Generalized Autoencoder

Autoencoder

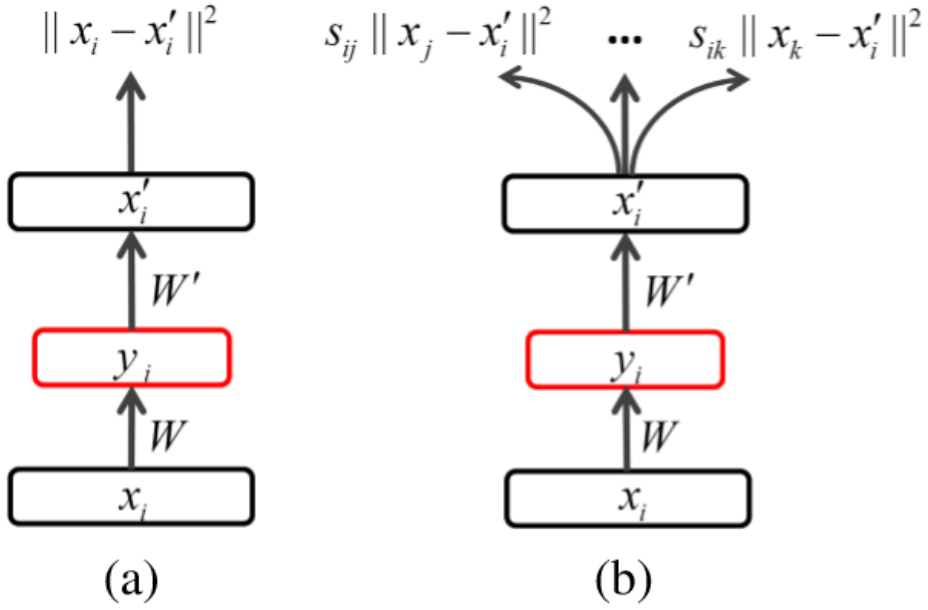
- The autoencoder algorithm belongs to a special family of dimensionality reduction methods implemented using artificial neural network.
- It tries to reconstruct the inputs at the outputs.
- It aims to learn a compressed representation for an input through minimizing its reconstruction error.



Autoencoder

In the **traditional autoencoder**, x_i is only used to reconstruct itself and the reconstruction error $\|x_i - x_i'\|^2$ just measures the distances between x_i and x_i'

In the **generalized autoencoder**, x_i involves in the reconstruction of a set of instances $\{x_j, x_k, \ldots\}$.
Each reconstruction error $s_{ij}\|x_j - x_i'\|^2$ measures a weighted distance between x_j and x_i'



Traditional autoencoder

Generalized autoencoder

Generalized Autoencoder

The encoder maps an input $x_i \in R^{d_x}$ to a reduced hidden representation $y_i \in R^{d_y}$ by a function $g()$

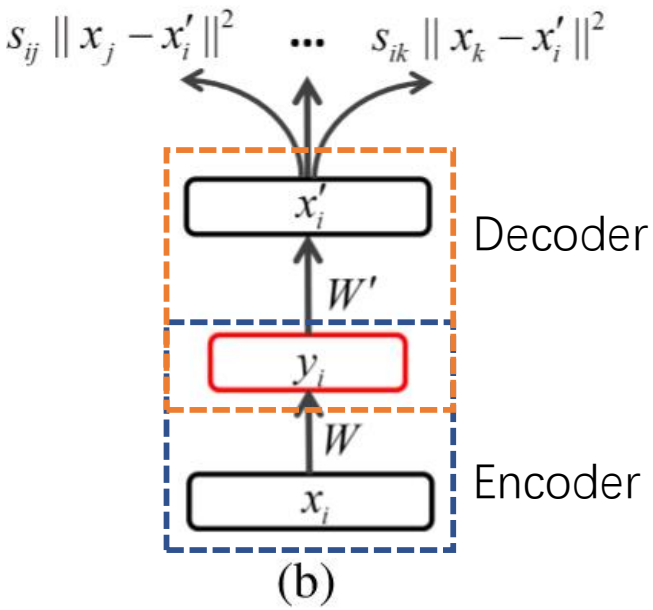
$$y_i = g(Wx_i)$$

where $g()$ is either the identity function (恒等函数) for a linear projection or a sigmoid function for a nonlinear mapping. The parameter W is a $d_y \times d_x$ weight matrix.

The decoder reconstructs $x_i' \in R^{d_x}$ from the hidden representation y_i

$$x_i' = f(W'y_i)$$

the parameter W' is another $d_x \times d_y$ weight matrix, which can be W^T



To model the relation between data, the decoder reconstructs a set of instances indexed by $\Omega_i = \{j, k, \ldots\}$ with specific weights $S_i = \{s_{ij}, s_{ik}, \ldots\}$

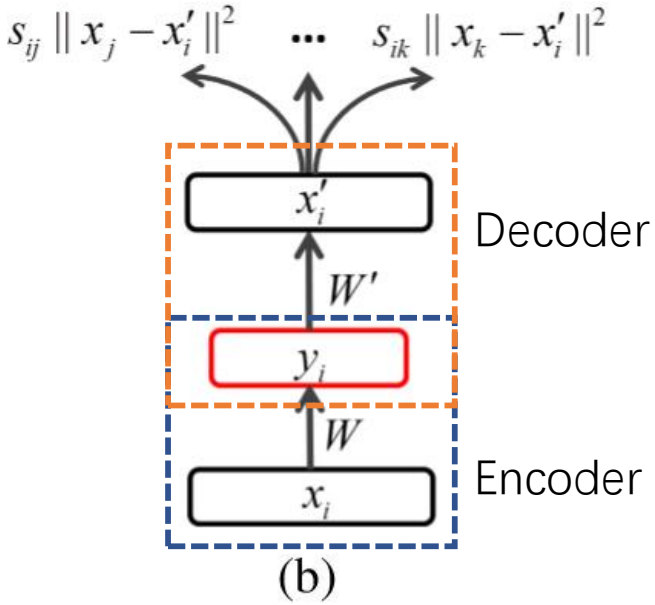
The weighted reconstruction error is

$$e_i(W, W') = \sum_{j \in \Omega_i} s_{ij} L(x_j, x_i')$$

The total reconstruction error $E\ of\ n$ sample is

$$E(W, W') = \sum_{i=1}^{n} e_i(W, W') = \sum_{i=1}^{n} \sum_{j \in \Omega_i} s_{ij} L(x_j, x_i')$$

Generalized Autoencoder



(b)

Algorithm 1 Iterative learning procedure for Generalized Autoencoder

Input: training set $\{x_i\}_1^n$

Parameters: $\Theta = (W, W')$

Notation: Ω_i: reconstruction set for x_i

S_i: the set of reconstruction weight for x_i

$\{y_i\}_1^n$: hidden representation

1. Compute the reconstruction weights S_i from $\{x_i\}_1^n$ and determine the reconstruction set Ω_i, e.g. by k-nearest neighbor
2. Minimize E in Eqn.4 using the stochastic gradient descent and update Θ for t steps
3. Compute the hidden representation $\{y_i\}_1^n$, and update S_i and Ω_i from $\{y_i\}_1^n$.
4. Repeat step 2 and 3 until convergence.

Connection to Graph Embedding

Graph embedding is known to be a general framework for dimensionality reduction, of which each data is represented as graph mode in a low-dimensional vector, the edges preserve similarities between data pairs.

$$y^* = arg \min_{y^T B y = c} \sum_{i,j} s_{ij} \left\| y_i - y_j \right\|^2$$

where y is the low-dimensional representation. s_{ij} is the similarity between the vertex i and j. c is a constant and B is constraint matrix.

The linear graph embedding (LGE) assumes that low-dimensional representation can be obtained by a linear projection $y = X^T w$, where w is the projection vector and $X=[x_1, x_2 \cdots, x_n]$. The objection function of LGF is

$$w^* = arg \min_{\substack{w^T X B X w = c \\ or w^T w = c}} \sum_{i,j} s_{ij} \left\| w^T x_i - w^T x_j \right\|^2$$

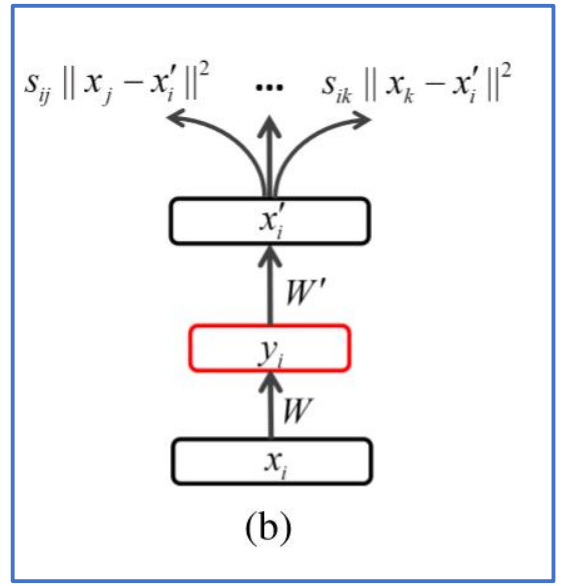
Connection to Graph Embedding

The total reconstruction error :

$$E(W, W') = \sum_{i=1}^{n} e_i(W, W') = \sum_{i=1}^{n} \sum_{j \in \Omega_i} s_{ij} L(x_j, x_i') \qquad (4)$$

Objective function in generalized autoencoder :

$$(W, W') = argmin \sum_{i,j} s_{ij} \left\| x_j - f(W' g(W x_i)) \right\|^2) \qquad (7)$$



In the linear reconstruction case, if $W' = W^T$, and assuming only one hidden node in the network, W degenerates to a column vector w , the objective function (7) becomes

$$w^* = \arg\min \sum_{i,j} s_{ij} ||x_j - w w^T x_i||^2 \qquad (8)$$

Let $w^T w = c$, and $y_i = w^T x_i$, Eqn. 8 becomes

$$w^* = \arg \min_{w^T w = c} \sum_{i,j} s_{ij} (||w^T x_i - w^T x_j||^2 + (\frac{c}{2} - 1) y_i^2) \qquad (9)$$

Connection to Graph Embedding

$$w^* = \arg\min \sum_{i,j} s_{ij} ||x_j - ww^T x_i||^2 \quad (8)$$

$$w^* = \arg \min_{w^T w = c} \sum_{i,j} s_{ij} (||w^T x_i - w^T x_j||^2 + (\frac{c}{2} - 1) y_i^2) \quad (9)$$

$$\|x_j - ww^T x_i\|^2$$

$$= x_j^T x_j + x_i^T ww^T ww^T x_i - 2x_j^T ww^T x_i$$

$$= x_j^T x_j + cx_i^T ww^T x_i - 2x_j^T ww^T x_i \quad (A)$$

$$\|w^T x_i - w^T x_j\|^2$$

$$= x_i^T ww^T x_i + x_j^T ww^T x_j - 2x_i^T ww^T x_j \quad (B)$$

(A) – (B) $= x_j^T x_j - x_j^T ww^T x_j + (c-1) x_i^T ww^T x_i$

$$= x_j^T x_j - x_j^T ww^T x_j + (c-1) x_i^T ww^T x_i$$

$$= x_j^T x_j - y_j^2 + (c-1) y_i^2$$

Implementation of the GAE Variants

As can be seen from the above analysis, the generalized autoencoder can also be a general framework for dimensionality reduction through defining different reconstruction sets and weights.

Table 1. Six implementations of the generalized autoencoders inspired by PCA [10], LDA [2], ISOMAP [15], LLE [12], LE [1] and MFA [19]

Method	Reconstruction Set	Reconstruction Weight
GAE-PCA	$j = i$	$s_{ij} = 1$
GAE-LDA	$j \in \Omega_{c_i}$	$s_{ij} = \frac{1}{n_{c_i}}$
GAE-ISOMAP	$j : x_j \in X$	$s_{ij} \in S = -H\Lambda H/2$
GAE-LLE	$j \in N_k(i)$,	$s_{ij} = (M + M^T - M^T M)_{ij}$ if $i \neq j$;
	$j \in (N_k(m) \cup m), j \neq i$ if $\forall m, i \in N_k(m)$	0 otherwise
GAE-LE	$j \in N_k(i)$	$s_{ij} = \exp\{-\|\|x_i - x_j\|\|^2/t\}$
GAE-MFA	$j \in \Omega_{k_1}(c_i)$,	$s_{ij} = 1$
	$j \in \Omega_{k_2}(\bar{c}_i)$	$s_{ij} = -1$

Deep Generalized Autoencoder

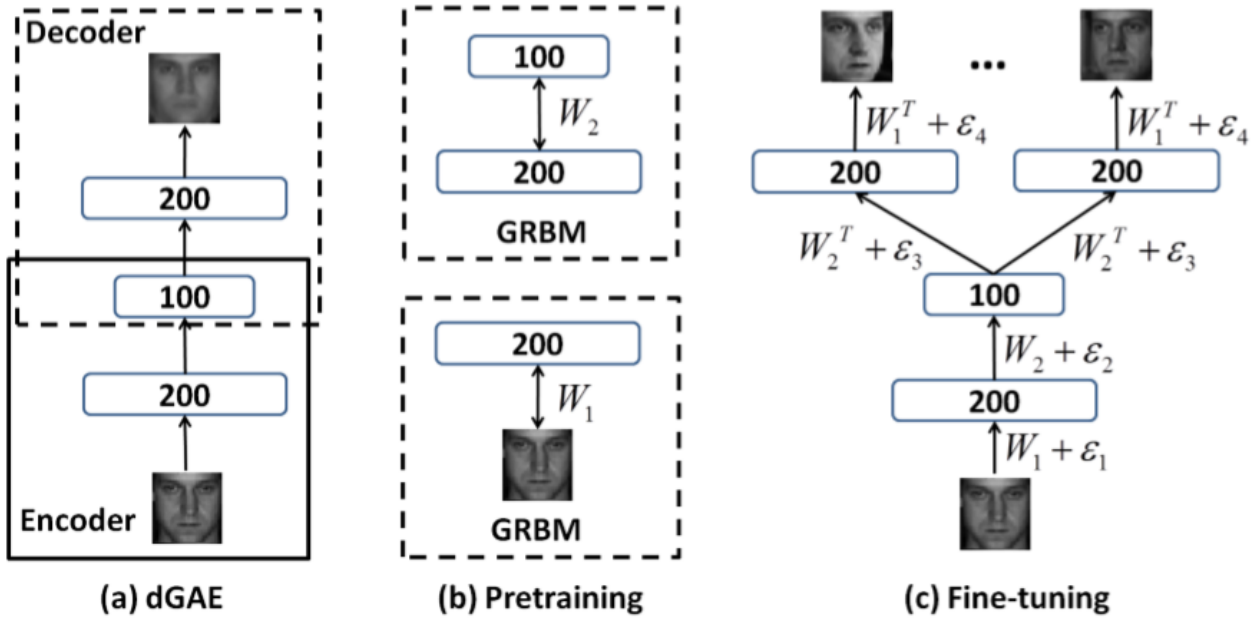


Figure 2. Training a deep generalized autoencoder (dGAE) on the face dataset used in Section 3.3. (a) The dGAE consists of a three-layer encoder network and a three-layer decoder network. A reconstructed face is in the output layer. (b) Pretraining the dGAE through learning a stack of Gaussian restricted Boltzmann machines (GRBM). (c) Fine-tuning a deep GAE-LDA.

"With large initial weights, autoencoders typically find poor local minima. ··· If the initial weights are close to a good solution, gradient descent works well. ··· We introduce this pretraining procedure for binary data, generalize it to real"

—— *G.E. Hinton "Reducing the Dimensionality of Data with Neural Networks "2006*

Experimental Results

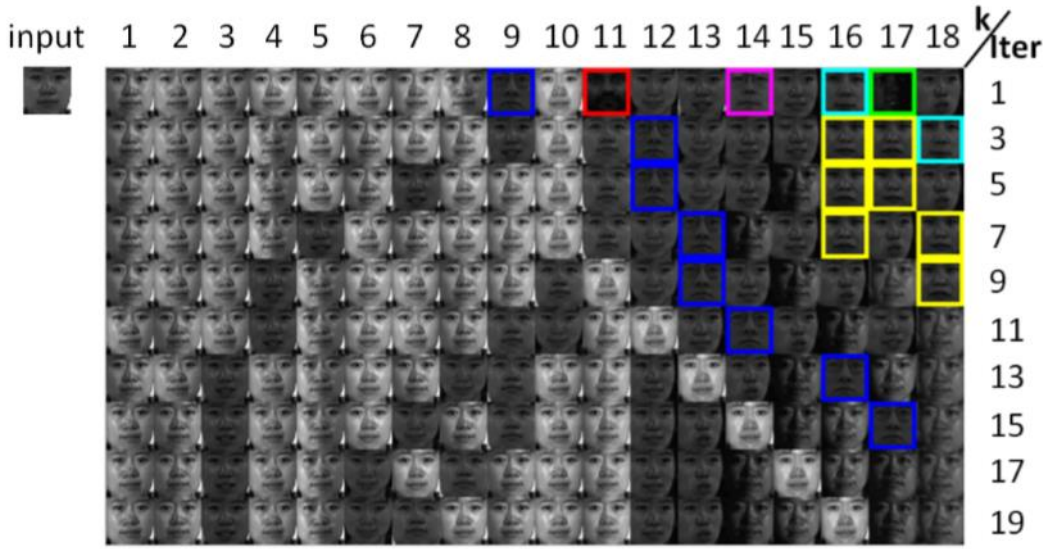


Figure 3. The changes of nearest neighbors during the iterative learning. Each row shows the 18 nearest neighbors of an input data during in one iteration of learning. The faces in the colored boxes belong to other persons. The color indicates the identity of the persons. We can see that along the iterative learning, the nearest neighbors of a data converges to its own class.

Fig 3 shows the changes of a face's 18 nearest neighbors during the first 20 iterations of our unsupervised dGAE-LE learning on the CMU PIE dataset

Experimental Results

CMU PIE dataset

Table 2. Performance comparison on the CMU PIE dataset. ER is short for "error rate". The reduced dimensions are in parentheses. Our models use 100 dimensions. g and pp are short for "Gaussian" and "polyplus" kernels.

Method	ER	Our Model	ER
PCA	20.6% (150)	dGAE-PCA	3.5%
Kernel PCA	8.1% (g)		
LDA	5.7% (67)	dGAE-LDA	1.2%
Kernel LDA	1.6% (pp)		
ISOMAP	–	dGAE-ISOMAP	2.5%
LLE	–	dGAE-LLE	3.6%
LPP	4.6%(110)	dGAE-LE	1.1%
Kernel LPP	1.7% (pp)		
MFA	2.6% (85)	dGAE-MFA	1.1%
Kernel MFA	2.1% (pp)		

MNIST dataset

Table 3. Performance comparison on the MNIST dataset. ER is short for "error rate". The reduced dimensions are in the parentheses. Our models use 30 dimensions. pp is short for "polyplus" kernel.)

Method	ER	Our Model	ER
PCA	6.2% (55)	dGAE-PCA	5.3%
Kernel PCA	8.5% (pp)		
LDA	16.1% (9)	dGAE-LDA	4.4%
Kernel LDA	4.6% (pp)		
ISOMAP	–	dGAE-ISOMAP	6.4%
LLE	–	dGAE-LLE	5.7%
LPP	7.9%(55)	dGAE-LE	4.3%
Kernel LPP	4.9% (pp)		
MFA	9.5% (45)	dGAE-MFA	3.9%
Kernel MFA	6.8% (pp)		

LPP is a popular linear approximation to the LE
"Locality preserving projections."

Experimental Results

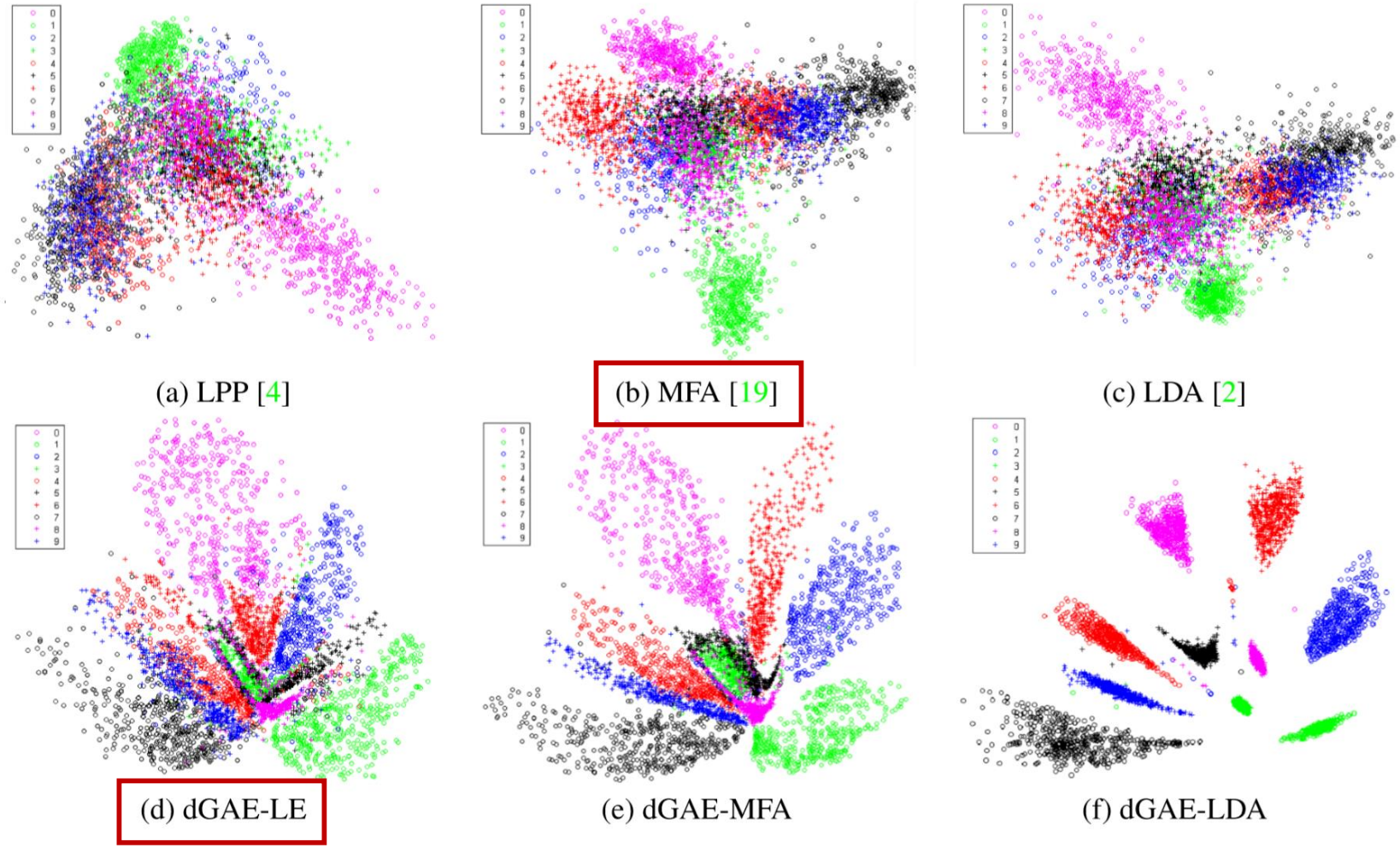


Figure 6. 2D visualization of the learned digit image manifold.

Thank You!