常用降维算法及原理

一． 基本概念

降维，通过单幅图像数据的高维化，将单幅图像转化为高维空间中的数据集合，对其进行非线性降维。寻求其高维数据流形本征结构的一维表示向量，将其作为图像数据的特征表达向量

降维的意思是能够用一组个数为 d 的向量$\mathbf{z_i}$来代表个数为 D 的向量$\mathbf{x_i}$所包含的有用信息，其中 d<D。假设对一张 512*512 大小的图片，用 svm 来做分类，最直接的做法是将图按照行或者列展开变成长度为 512*512 的输入向量$\mathbf{x_i}$，跟 svm 的参数相乘。假如能够将 512*512 的向量在保留有用信息的情况下降维到 100，那么存储输入和参数的空间会减少很多，计算向量乘法的时间也会减少很多。所以降维能够有效的减少计算时间。而高维空间的数据很有可能出现分布稀疏的情况，即 100 个样本在 100 维空间分布肯定是非常稀疏的，每增加一维所需的样本个数呈指数级增长，这种在高维空间中样本稀疏的问题被称为维数灾难。降维可以缓解这种问题。

而为什么可以降维，这是因为数据有冗余，要么是一些没有用的信息，要么是一些重复表达的信息，例如一张 512*512 的图只有中心 100*100 的区域内有非 0 值，剩下的区域就是没有用的信息，又或者一张图是成中心对称的，那么对称的部分信息就重复了。正确降维后的数据一般保留了原始数据的大部分的重要信息，它完全可以替代输入去做一些其他的工作，从而很大程度上可以减少计算量。例如降到二维或者三维来可视化

二． 常用降维算法

主成分分析 PCA

PCA 由 Karl Pearson 在 1901 年发明，是一种线性降维方法，高维空间(维数为 D)的某个点$\mathbf{x_i} = (x_1, x_2, \ldots, x_D)$通过与矩阵 W 相乘映射到低维空间(维数为 d，*d*<*D*)中的某个点$\mathbf{z_i} = W^T \mathbf{x_i}$，其中 W 的大小是 *D***d*，i 对应的是第 i 个样本点。从而可以得到 N 个从 D 维空间映射到 d 维空间的点，PCA 的目标是让映射得到的点$\mathbf{z_i}$尽可能的分

开，即让 N 个$\mathbf{z_i}$的方差尽可能大。假如 D 维空间中的数据每一维均值为 0，即$\sum_i \mathbf{x_i} = \mathbf{0}$，那么两边乘上$W^T$得到的降维后的数据每一维均值也是 0，考虑一个矩阵$C = \frac{1}{N} X * X^T$，这个矩阵是这组 D 维数据的协方差矩阵，可以看出对角线上的值是 D 维中的某一维内的方差，非对角线元素是 D 维中两维之间的协方差。

$$\frac{1}{N} X * X^T = \begin{pmatrix} \frac{1}{N}\sum_{i=1}^{N} x_{1i}^2 & \cdots & \frac{1}{N}\sum_{i=1}^{N} x_{1i}^T x_{Di} & \cdots \\ \cdots & \frac{1}{N}\sum_{i=1}^{N} x_{Di}^T x_{1i} & \cdots & \frac{1}{N}\sum_{i=1}^{N} x_{Di}^2 \end{pmatrix}$$

那么针对降维后 d 维数据的协方差矩阵$B = \frac{1}{N} Z * Z^T$，如果希望降维后的点尽可能分开，那么就希望 B 对角线上值即每一维的方差尽可能大，方差大说明这些维上数据具有非常好的区分性，同时希望 d 的每一维都是正交的，它们正交就会使得两个维是无关的，那么它们就不会包含重叠的信息，这样就能最好的表现数据，每一维都具有足够的区分性，同时还具有不同的信息。这种情况下 B 非对角线上值全部为 0。又由于可以推导得出

$$B = \frac{1}{N} Z * Z^T = W^T * (\frac{1}{N} X * X^T) W = W^T * C * W$$

这个式子实际上就是表示了线性变换矩阵 W 在 PCA 算法中的作用是让原始协方差矩阵 C 对角化。又由于线性代数中对角化是通过求解特征值与对应的特征向量得到，因此可以推出 PCA 算法流程(流程主要摘自周志华老师的《机器学习》一书，其中加入了目标和假设用于对比后面的算法。周老师书中是基于拉格朗日乘子法推导出来，本质上而言与[3]都是一样的，这里很推荐这篇讲 PCA 数学原理的博客[3])。

输入:N 个 D 维向量$\mathbf{x_1}, \ldots, \mathbf{x_N}$，降维到 d 维

输出:投影矩阵$W = (w_1, \ldots, w_d)$，其中每一个w_i都是 D 维列向量

目标:投影降维后数据尽可能分开，$\max_w tr(W^T X X^T W)$(这里的迹是因为上面提到的 B 的非对角线元素都是 0，而对角线上的元素恰好都是每一维的方差)

假设:降维后数据每一维方差尽可能大，并且每一维都正交

1.将输入的每一维均值都变为 0，去中心化

2.计算输入的协方差矩阵$C = X * X^T$

3.对协方差矩阵 C 做特征值分解

4.取最大的前 d 个特征值对应的特征向量$w_1, \ldots, w_d$

此外，PCA 还有很多变种 kernel PCA, probabilistic PCA 等等，本文暂时只考虑最简单的 PCA 版本

多维缩放(MDS)

MDS 的目标是在降维的过程中将数据的 dissimilarity(差异性)保持下来，也可以理解降维让高维空间中的距离关系与低维空间中距离关系保持不变。这里的距离用矩阵表示，N 个样本的两两距离用矩阵 A 的每一项a_{ij}表示，并且假设在低维空间中的距离是欧式距离。而降维后的数据表示为$\mathbf{z_i}$，那么就有

$a_{ij} = |\mathbf{z_i} - \mathbf{z_j}|^2 = |\mathbf{z_i}|^2 + |\mathbf{z_j}|^2 - 2\mathbf{z_i}\mathbf{z_j}^T$。右边的三项统一用内积矩阵 E 来表示$e_{ij} = \mathbf{z_i}\mathbf{z_j}^T$。去中心化之后，E 的每一行每一列之和都是 0，从而可以推导得出

$$e_{ij} = -\frac{1}{2}(a_{ij}^2 - a_{i\cdot} - a_{\cdot j} - a_{\cdot\cdot}^2) = -\frac{1}{2}(e_{ii} + e_{jj} - 2e_{ij} - \frac{1}{N}(tr(E) + Ne_{jj}) - \frac{1}{N}(tr(E) + Ne_{ii}) + \frac{1}{N^2}(2Ne_{jj})) = e_{ij} = -\frac{1}{2}(PAP)_{ij}$$

其中$P = I - \frac{1}{N}\mathbf{1}$单位矩阵 I 减去全 1 矩阵的$\frac{1}{N}$，i· 与 ·j 是指某列或者某列总和，从而建立了距离矩阵 A 与内积矩阵 E 之间的关系。因而在知道 A 情况下就能够求解出 E，进而通过对 E 做特征值分解，令$E = V\Lambda V^T$，其中 Λ 是对角矩阵，每一项都是 E 的特征值 λ1≥…≥λd，那么在所有特征值下的数据就能表示成$Z = \Lambda^{\frac{1}{2}}V^T$，当选取 d 个最大特征值就能让在 d 维空间的距离矩阵近似高维空间 D 的距离矩阵，从 MDS 流程如下[2]:

输入:距离矩阵$A^{N*N} = a_{ij}$，上标表示矩阵大小，原始数据是 D 维，降维到 d 维

输出:降维后矩阵$Z^{d*N} = \widetilde{\Lambda}^{\frac{1}{2}}\widetilde{V}$

目标:降维的同时保证数据之间的相对关系不变

假设:已知 N 个样本的距离矩阵

1.算出$a{i\cdot}、a{\cdot j}、a_{\cdot \cdot}$

2.计算内积矩阵 E

3.对 E 做特征值分解

4.取 d 个最大特征值构成$\widetilde{V}$，对应的特征向量按序排列构成$\widetilde{\Lambda}$

线性判别分析(LDA)

LDA 最开始是作为解决二分类问题由 Fisher 在 1936 年提出，由于计算过程实际上对数据做了降维处理，因此也可用作监督线性降维。它通过将高维空间数据投影到低维空间，在低维空间中确定每个样本所属的类，这里考虑 K 个类的情况。它的目标是将样本能尽可能正确的分成 K 类，体现为同类样本投影点尽可能近，不同类样本点尽可能远，这点跟 PCA 就不一样，PCA 是希望所有样本在某一个维数上尽可能分开，LDA 的低维投影可能会重叠，但是 PCA 就不希望投影点重叠。它采用的降维思路跟 PCA 是一样的，都是通过矩阵乘法来进行线性降维，投影点是$\mathbf{z_i} = W^T * \mathbf{x_i}$。假设按下图中的方向来投影，投影中心对应的原来高维点分别是μ1，μ2。由于希望属于不同类的样本尽可能离的远，那么就希望投影过后的投影中心点离的尽可能远，即目标是

$\underset{W}{max}\ ||W^T\mu_1 - W^T\mu_2||^2$，但是仅仅有中心离的远还不够，例如下图中垂直于 x1 轴投影，两个中心离的足够远，但是有样本在投影空间重叠，因此还需要额外的优化目标，即同类样本投影点尽可能近。那么同类样本的类间协方差就应该尽可能小，同类样本的协方差矩阵如下。

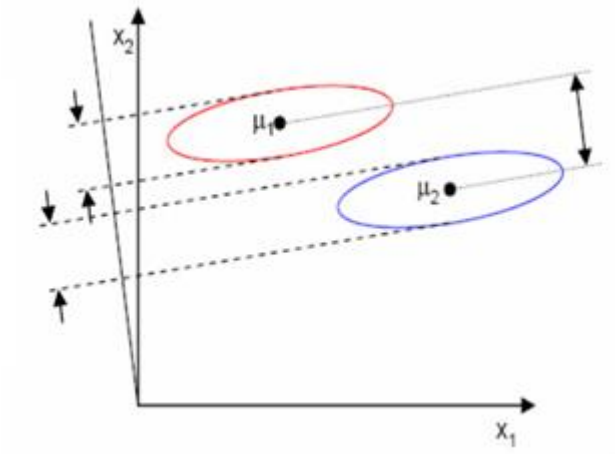

图 3 LDA 进行投影(图来源[4])

$$\underset{w}{min} = \sum_{X\in X_1}(W^TX - W^T\mu_1)(W^TX - W^T\mu_1)^T = W^T(\sum_{X\in X_1}(X-\mu_1)(X$$

其中 μ1=(u1,…,uN),W=(w1,…,wd)，X_1表示样本属于第 1 类的集合，中间的矩阵$\sum_{X_1}$是属于第X_1的样本协方差矩阵，将 K 个类的原始数据协方差矩阵加起来称为类内散度矩阵，$S_w = \sum_{k=1}^{K} S_k = \sum_{k=1}^{K}\sum_{X\in X_k}(X-\mu_k)(X-\mu_k)^T$。而上面两个类的中心距离是中心直接相减，K 个类投影中心距离需要先计算出全部样本的中

心$\mu = \frac{1}{N}\sum_{k=1}^{K} N_k\mu_k$($N_k$表示属于第 k 类的样本个数)，通过类间散度矩阵来衡量，即$S_b = \sum_{k=1}^{K} N_k(\mu_\mathbf{k} - \mu)(\mu_\mathbf{k} - \mu)^T$。整合一下，LDA 算法的优化目标是最大化下面的 costfunction:

$$\max_W J(W) = \frac{tr(W^T(\sum_{k=1}^{K} N_k(\mu_\mathbf{k} - \mu)(\mu_\mathbf{k} - \mu)^T)W)}{tr(W^T\sum_{k=1}^{K}\sum_{X\in X_k}(X - \mu_k)(X - \mu_k)^T)W)} = \frac{tr(W^T S_b W)}{tr(W^T S_w W)}$$

二分类情况下，W 的大小是 D*1，即 J(W)本身是个标量，针对 K 类的情况，W 的大小是 D*d−1，优化的目标是对上下的矩阵求它的迹。这里可以发现在 LDA 中没有对数据去中心化，如果要去中心化每个类的中心就会重叠了，所以这个算法没有去中心化。然后 J(W)对 W 求导，这个式子就表明了 W 的解是$S_w^{-1}S_b$的 d-1 个最大特征值对应的特征向量组成的矩阵。那么就可以通过 W 来对 X 进行降维。

$$\frac{\partial J(W)}{\partial W} = (W^T S_w W)2S_b W - (W^T S_b W)2S_w W = 0 \Rightarrow S_b W - J(W)S_w W = 0 \Rightarrow S_w^{-1}S_b W = J(W)W$$

输入:N 个 D 维向量$\mathbf{x_1}, \ldots, \mathbf{x_N}$，数据能够被分成 d 个类

输出:投影矩阵$W=(w1, ⋯, w{d-1})，其中每一个，其中每一个 w_i$都是 D 维列向量

目标:投影降维后同一类的样本之间协方差尽可能小，不同类之间中心距离尽可能远

假设:优化目标是最大化$\frac{tr(W^T S_b W)}{tr(W^T S_w W)}$

1.求出类内散度矩阵 Sw 和类间散度矩阵 Sb

2.对做奇异值分解$S_w = U\sum V^T$，求得$S_w^{-1} = V\sum^{-1}U^T$

3.对矩阵$S_w^{-1}S_b$做特征分解

4.取最大的前 d-1 个特征值对应的特征向量$w1,⋯,w{d-1}$

等度量映射(Isomap)

上面提到的 MDS 只是对数据降维，它需要已知高维空间中的距离关系，它并不能反

应出高维数据本身潜在的流形，但是可以结合流形学习的基本思想和 MDS 来进行降维[5]。也就是高维空间的局部空间的距离可以用欧式距离算出，针对 MDS 的距离矩阵 A，某两个相邻的点之间距离$A_{ij} = |x_i - x_j|$也就是它们的欧式距离，距离比较近的点则通过最短路径算法来确定，而离的比较远的两点之间 Aij = ∞，把矩阵 A 确定下来，那么这里就涉及到判断什么样的点相邻，Isomap 是通过 KNN 来确定相邻的点，整体算法流程如下:

输入:N 个 D 维向量$\mathbf{x_1}, \ldots, \mathbf{x_N}$，一个点有 K 个近邻点，映射到 d 维

输出:降维后矩阵$Z^{d*N} = \widetilde{\Lambda}^{\frac{1}{2}} \widetilde{V}$

目标:降维的同时保证高维数据的流形不变

假设:高维空间的局部区域上某两点距离可以由欧式距离算出

1.由 KNN 先构造 A 的一部分，即求出相邻的点并取它们的欧式距离填入 Aij，其他的位置全部初始化为无穷大

2.根据最短路径算法(Dijkstra 算法)找到距离比较近的点之间的路径并填入距离

3.将距离矩阵 A 作为 MDS 的输入，得到输出

局部线性嵌入(LLE)

如之前提到过的，流形学习的局部区域具有欧式空间的性质，那么在 LLE 中就假设某个点 xi 坐标可以由它周围的一些点的坐标线性组合求出，即$\mathbf{x_i} = \sum_{j \in X_i} f_{ij} \mathbf{x_j}$(其中 Xi 表示 xi 的邻域上点的集合)，这也是在高维空间的一种表示。由于这种关系在低维空间中也被保留，因此$\mathbf{z_i} = \sum_{j \in Z_i} f_{ij} \mathbf{z_j}$，两个式子里面权重取值是一样的。

基于上面的假设，首先想办法来求解这个权重，假设每个样本点由周围 K 个样本求出来，那么一个样本的线性组合权重大小应该是 1*K，通过最小化 reconstruct error 重构误差来求解，然后目标函数对 f 求导得到解。

$$\min_{f_1, \cdots, f_K} \sum_{k=1}^{K} \left| \mathbf{x_i} - \sum_{j \in X_i} f_{ij} \mathbf{x_j} \right| \qquad s.t. \sum_{j \in X_i} f_{ij} = 1$$

求出权重之后，代入低维空间的优化目标

$$\min_{z_1,\cdots,z_K} \sum_{k=1}^{K} \left|\mathbf{z_i} - \sum_{j\in Z_i} f_{ij}\mathbf{z_j}\right| = tr((Z - Z*F)(Z - Z*F)^T) = tr(ZM$$

来求解 Z，这里将 F 按照 N*K 排列起来，且加入了对 Z 的限制。这里用拉格朗日乘子法可以得到 MZ=λY 的形式，从而通过对 M 进行特征值分解求得 Z。

输入:N 个 D 维向量$\mathbf{x_1},\ldots,\mathbf{x_N}$，一个点有 K 个近邻点，映射到 d 维

输出:降维后矩阵 Z

目标:降维的同时保证高维数据的流形不变

假设:高维空间的局部区域上某一点是相邻 K 个点的线性组合，低维空间各维正交

1.由 KNN 先构造 A 的一部分，即求出 K 个相邻的点，然后求出矩阵 F 和 M

2.对 M 进行特征值分解

3.取前 d 个非 0 最小的特征值对应的特征向量构成 Z(这里因为最小化目标，所以取小的特征值)

t-SNE

t-SNE 也是一种将高维数据降维到二维或者三维空间的方法，它是 2008 年由 Maaten 提出[6]，基于 2002 年 Hinton 提出的随机近邻嵌入(StochasticNeighbor Embedding, SNE)方法的改进。主要的思想是假设高维空间中的任意两个点，xj 的取值服从以 xi 为中心方差为 σi 的高斯分布，同样 xi 服从以 xj 为中心方差为 σj 的高斯分布,这样 xj 与 xi 相似的条件概率就为

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

即 xj 在 xi 高斯分布下的概率占全部样本在 xi 高斯分布下概率的多少，说明了从 xi 角度来看两者的相似程度。接着令 pij =(pi|j+pj|i)/2n 用这个概率来作为两个点相似度在全部样本两两相似度的联合概率 pij 。公式如下，论文没有解释 σ 是标量还是矢量，但是因为在后续的求解中 pij 不是直接由下面这个联合概率公式求出，而是通过前面的条件概率来求，前面的式子针对每一个样本 i 都会计算一个 σi，具体给定一个确定值 $Prep(P_i) = 2^{H(P_i)}$，其中$H(P_i) = -\sum_j p_{j|i} log p_{j|i}$。接着通过二分查找来确定 xi 对应的 σi，使得代入上面的两个式子后等于 Prep 的值，因此这里的 σ 应该是个矢量。不太可能所有样本都共用一个高斯分布参数。

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

同时将低维空间两个点的相互关系或者说相似程度也用联合概率来表示，假设在低维空间中两点间欧式距离服从一个自由度的学生 t 分布，那么在低维空间中两个点的距离概率在所有的两个点距离概率之中的比重作为它们的联合概率。

$$q_{ij} = \frac{(1+|\mathbf{z_i}-\mathbf{z_j}|^2)^{-1}}{\sum_{k\neq l}(1+|\mathbf{z_k}-\mathbf{z_l}|^2)^{-1}}$$

假如在高维空间的 xi,xj 与对应在低维空间中的 zi,zj 算出来的相似性值 pij,qij 相等，那么就说明低维空间的点能够正确的反应高维空间中的相对位置关系。所以 tsne 的目的就是找到一组降维表示能够最小化 pij 和 qij 之间的差值。因此，tsne 采用了 KullbackLeibler divergence 即 KL 散度来构建目标函数 $J = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}}$，KL 散度能够用来衡量两个概率分布的差别。它通过梯度下降的方法来求输入数据对应的低维表达 zi，即用目标函数对 zi 求导，把 zi 作为可优化变量，求得每次对 zi 的梯度为 $\frac{\partial J}{\partial \mathbf{z_i}} = 4\sum_j (p_{ij}-q_{ij})(1+||\mathbf{z_i}-\mathbf{z_j}||^2)^{-1}$，然后更新迭代 zi，在实际更新的过程中则像神经网络的更新一样加入了 momentum 项为了加速优化，大概的算法流程如下:

输入:N 个 D 维向量 $\mathbf{x_1},\ldots,\mathbf{x_N}$，映射到二维或者三维，定值 Perp，迭代次数 T，学习率 η，momentum 项系数α(t)

输出:降维后数据表示 z1,…,zN

目标:降维到二维或者三维可视化(重点是可视化)

假设:在高维空间中，一个点 xj 的取值服从以另外一个点 xi 为中心的高斯分布。在低维空间中，两个点之间的欧式距离服从自由度为 1 的 t 分布

1.先由二分查找确定 xi 的 σi

2.计算成对的$P{j|i}，得到，得到 p{ij} = (p{j|i}+p{i|j})/2$

3.初始化 z1,…,zN

4.计算 qij

5.计算梯度 ∂J/∂zi

6.更新

$$\mathbf{z_i}^{(t)} = \mathbf{z_i}^{(t-1)} + \eta\frac{\partial J}{\partial \mathbf{z_i}} + \alpha(t)(\mathbf{z_i}^{(t-1)} - \mathbf{z_i}^{(t-2)})$$

7.重复 4~6 至收敛或者完成迭代次数 T

需要注意的是，这个算法将低维数据作为变量进行迭代，所以如果需要加入插入新的数据，是没有办法直接对新数据进行操作，而是要把新数据加到原始数据中再重新算一遍，因此 T-sne 主要的功能还是可视化

DeepAutoencoder Networks

Autoencoder 是神经网络的一种，它是一种无监督算法，可以用来降维也能用来从数据中自动学习某种特征，这个神经网络的原理是输入一组值，经过网络之后能够获得一组输出，这组输出的值尽可能的跟输入的值大小一致。网络由全连接层组成，每层每个节点都跟上一层的所有节点连接。Autoencoder 的结构如下图 4 所示，encoder 网络是正常的神经网络前向传播 z=W x+b，decoder 网络的传播参数是跟它成对称结构的层参数的转置，经过这个网络的值为$\mathbf{x'} = \mathbf{W^T z} + \mathbf{b^T}$，最后传播到跟网络的输入层个数相等的层时，得到一组值 x′，网络希望这两个值相等 x′=x，这个值与真实输入 x 值通过交叉熵或者均方误差得到重构误差的 costfunction，再通过最小化这个 cost 和梯度下降的方法使网络学到正确的参数。因此可以通过这个网络先经过"encoder"网络将高维数据投影到低维空间，再经过"decoder"网络反向将低维数据还原到高维空间。

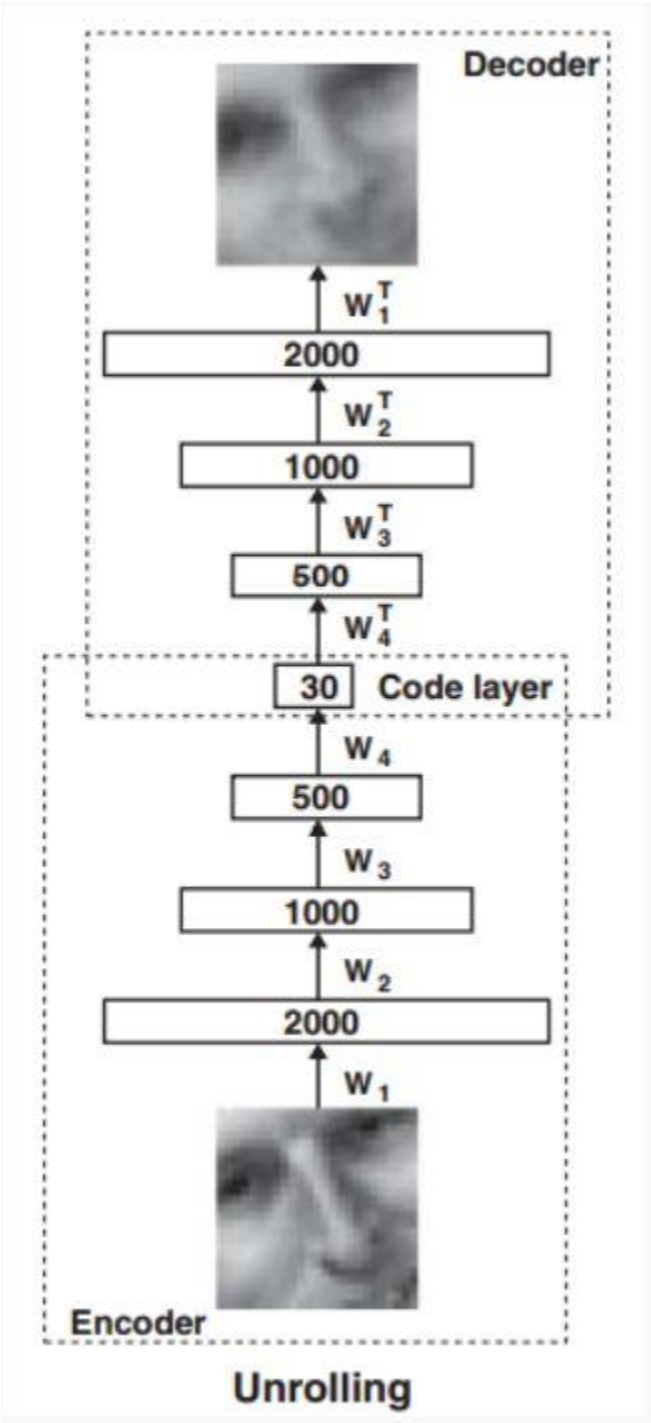

图 4 Autoencoder 网络结构图

然而在实际的实现网络过程中，整个网络实际上层数只是图 4 中的一半，即 4 层网络，2000-1000-500-30 的全连接结构。因为权重参数实际上在 encoder 和 decoder 中是相同的，enocoder 过程是上一层的节点值乘以权重得到这一层的节点值，而 decoder 是这一层节点值与权重矩阵的转置相乘得到上一层的节点值。下图[7]更加清晰的展示了每一层实际的结构，包括一次前向传播和后向传播，因此可以拿最顶层的值作为网络的降维输出来进行其他的分析，例如可视化，或者作为压缩特征使用。

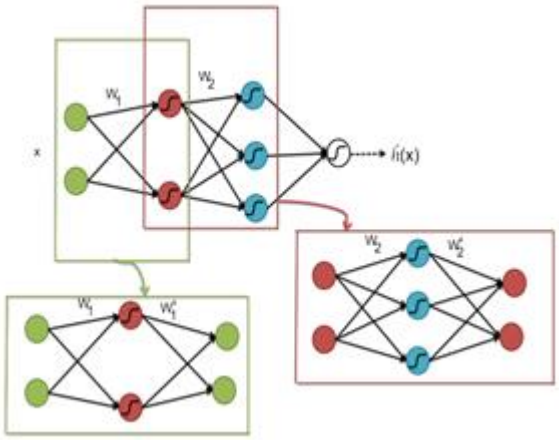

图 5 Autoencoder 层间结构

06 年的时候 Hinton 在 science 上发了一篇文章讲如何用深度学习中的 autoencoder 网络来做降维[8]，主要是提出了先通过多层 RBM 来预训练权重参数，用来解决 autoencoder 降维后的质量依赖初始化网络权重的问题，即主要目的是提出一种有效的初始化权重的方式。上面的表达式中没有加入非线性变换，真实网络中每一层跟权重做矩阵乘法之后还需要加上非线性变换。此外，autoencoder 的模型中可以加入 sparsity 的性质[9]，即针对 N 个 D 维输入，某一层的某一个节点输出值之和$\widehat{\rho_j}^{(l)}$趋近于 0，即$\widehat{\rho_j}^{(l)} = \frac{1}{N}\sum_{i=1}^{N}[a_j^{(l)}(\mathbf{x^{(i)}})]$，其中 l 代表是哪一层，i 代表是第几个输入。也能加对权重有要求的正则项。

输入:N 个 D 维向量 x1,…,xN，网络结构即每层节点数，迭代次数 T，学习率 η
输出:降维后数据表示 z1,…,zN
目标:网络能够学习到数据内部的一些性质或者结构，从而能够重构输入数据
假设:神经网络就是特牛逼，就是能学到特征，科科
1.设置层数和每一层节点数
2.初始化权重参数
3.前向传播计算下一层的节点值 z=W x+b
4.反向传播计算上一层反向节点值$\mathbf{x'} = \mathbf{W^T z} + \mathbf{b^T}$
5.计算每一层对输入和对这层参数 W 的梯度，利用反向传播将 error 传递到整个网络
6.将分别对 W 和W^T的梯度求和然后更新 W
7.重复 3~6 至收敛或者完成迭代次数 T