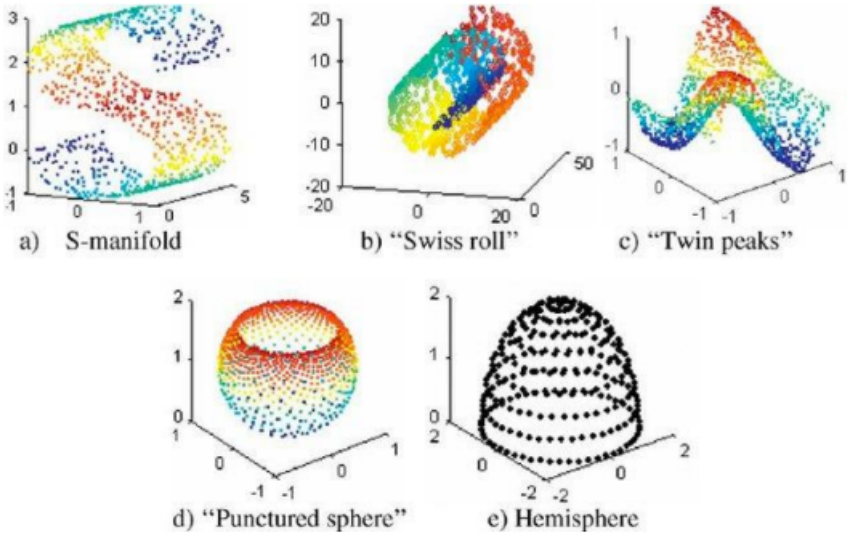
Lecture7 PCA & Autoencoders

1. 流形假设 Manifold hypothesis

介绍

现实世界的数据生活在一个低维非线性流形中

- 现实中的对事物可以参考一些具有局部相似性的数字
- 换句话说，数据以高概率集中在一个**小的非线性空间区域**
- 数据集中在低维流形上



- 为什么要研究流形的性质
    - 如果给定的输入来自于流形，一些 ML 算法会有不寻常的行为
- 自动编码器进一步采取这一想法，旨在学习流形的结构

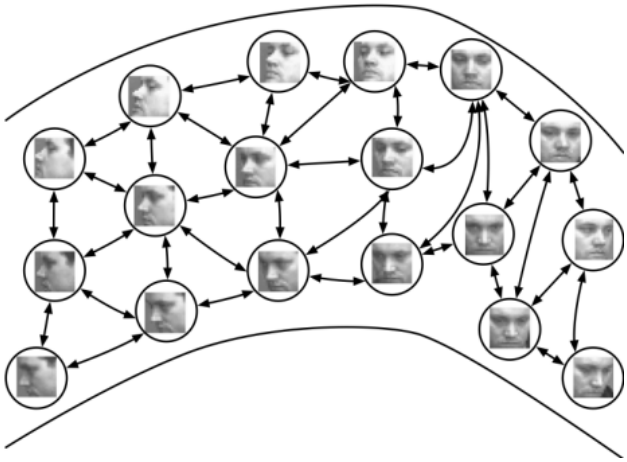
流形假设的本质

- 样本的局部相似度

示例

- QMUL 脸部数据集：133张面部图像覆盖偏航+/-90度和倾斜+/-30度，以10度为增量
- 人脸大约有50块肌肉，3个直角坐标(平移)和3个欧拉角(旋转)，所以一个人的面部流形不超过56个维度

如何自动找到这些尺寸/坐标?



- 可以提供脸与脸角度之间的不同关系
- 生成这样一种流形来预测新的情况，通过插值来寻找
- 如何提取流形特征有两种方法
    - 线性方法：PCA 主成分分析
    - 非线性方法：自动编码器

为什么数据会保存在流形上

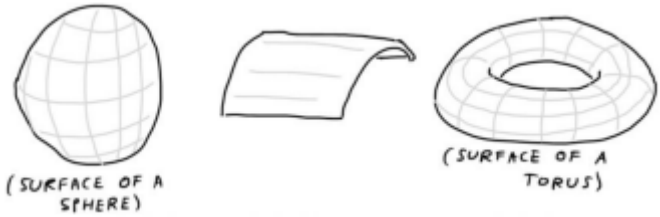
- 假设我们想对所有像素为 m x n 的(b&w)图像进行分类
    - 每个像素都有一个数值
    - 图像是一个 N = mn 维的单点
- 假设所有 m x n 个图像都是爱因斯坦的照片
    - 我们在像素值的选择上受到限制
    - 随机选择不会产生这样的图像
    - 因此，我们期望有更少的选择自由

流形假设定义

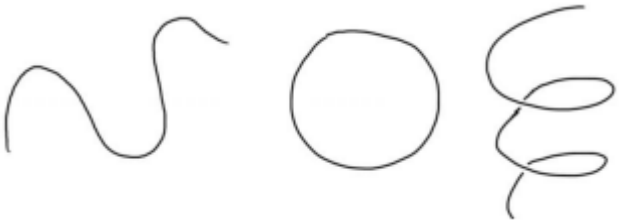
- 这个子集应该存在于低维的(环境)空间，实际上比输入图像的维度小很多很多
- 低维结构产生于物理定律的约束
- 经验研究
  - 大量的 3 x 3 的图片用来表达一个在 R^9 上的点
  - 一般依赖于一个 2D 的流形，被称为克莱因瓶

流形具有尺寸

- 二维流形是一个曲面



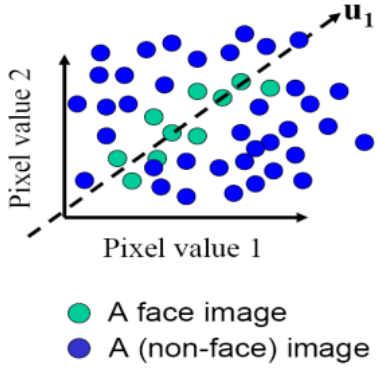
- 一维流形是一条曲线



- 零维流形是一个点
- 所有的三维空间 R^3 是一种三维流形

2. PCA 主成分分析

介绍



- 目标是找到最能解释数据**变化(全局)**的方向
- 找到几个方向(子空间)，使子空间中投影数据的方差**最大化**
- 给定一个在 R^d 维度内的数据点 $x_1, \ldots, x_N$
- 在 R^d 中选择一个单位向量 $\mathbf{u}$，它能捕捉到最大的数据**方差**
- 因此，如果我们找到了这样一个向量，我们可以通过将数据投影到这些方向来构造一组新的特征

$$u\left(\mathbf{x}_i\right)=\mathbf{u}^T\left(\mathbf{x}_i-\boldsymbol{\mu}\right)$$

- $\boldsymbol{\mu}$：一组数据点的的均值
- 我们想要把 $\mathbf{x}_i$ 投影到向量 $\mathbf{u}$ 上，计算公式为 $\frac{\mathbf{x}_i^T\mathbf{u}}{||\mathbf{u}||}$

计算方向向量

- 使投影数据方差最大化的方向

目标：最大化下面的公式

$$\frac{1}{N}\sum_{i=1}^{N}\underbrace{\mathbf{u}^{\mathrm{T}}\left(\mathbf{x}_i-\mu\right)}_{\text{Projection of data point}}\left(\mathbf{u}^{\mathrm{T}}\left(\mathbf{x}_i-\mu\right)\right)^{\mathrm{T}}$$

- 其中 $||\mathbf{u}|| = 1$
- $\mu = \frac{1}{N}\sum \mathbf{x}_i$

简化为如下

$$\begin{aligned}&=\mathbf{u}^{\mathrm{T}}\underbrace{\left[\frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{x}_i-\mu\right)\left(\mathbf{x}_i-\mu\right)^{\mathrm{T}}\right]}_{\text{Covariance matrix of data}}\mathbf{u}\\&=\mathbf{u}^{\mathrm{T}}\mathbf{M}_{\text{Covariance Matrix}}\mathbf{u}\end{aligned}$$

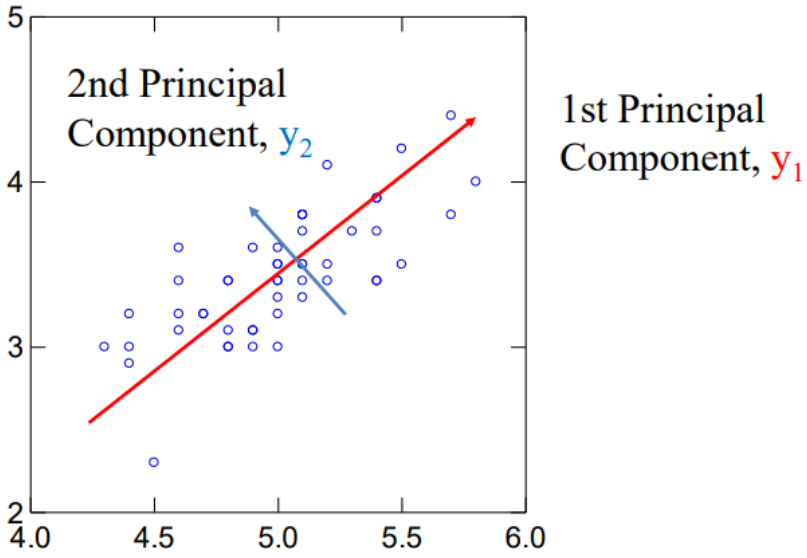
- 所以，我们想要最大化上面这个公式

实现问题

- 协方差矩阵很大，如果有 d 维，那么协方差矩阵有 d^2 维
- 通常情况下，训练样本的数量 N << 样本维度 d
- 一些简单的小技巧
  - 假设我们已经有一个 $\mathbf{X}$ 矩阵
    - 它是 $N \times d$ 维度的，每一行是一个已经中心化好的样本 $x_i - \mu$
    - 一共有 N 个样本
  - 可以使用 $\mathbf{X}\mathbf{X}^{\mathbf{T}}$（是一个 $N \times N$ 维的向量）求解特征向量 $\mathbf{u}$
  - 然后 $\mathbf{X}^{\mathbf{T}}\mathbf{u}$ 就是 $\mathbf{X}^{\mathbf{T}}\mathbf{X}$ 的特征向量
  - 可能需要规范化 $\mathbf{u}$ (以获得单位长度向量)

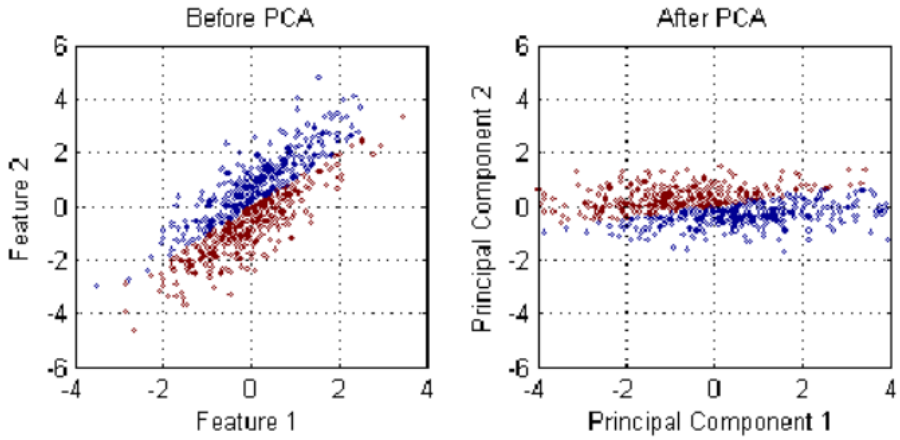
PCA 图解

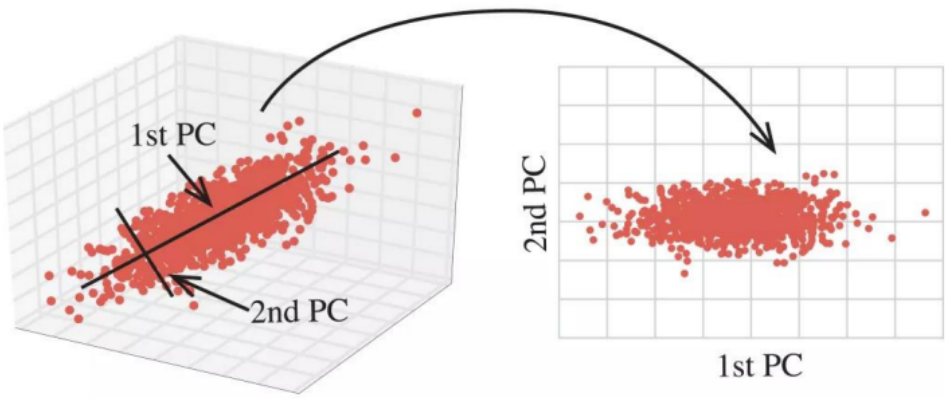
对于下图的示例，我们计算的两个特征向量应该如下

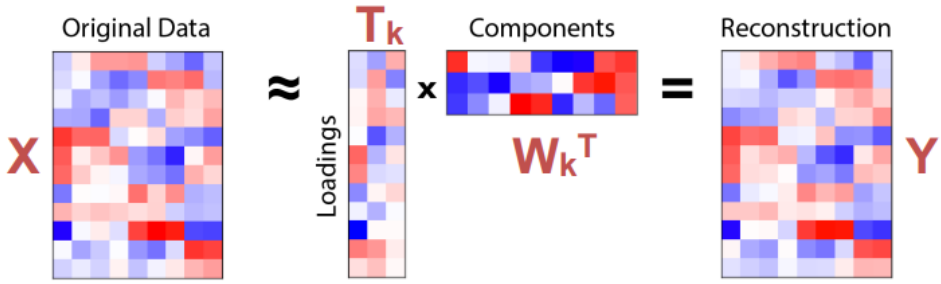


总结

- PCA定义了一个从输入 $\mathbf{x}$ 到表示的**正交线性变换** $z = f(x)$
- 正式地说，如果 $\mathbf{X}$ （$N \times d$）是每一行对应一个数据样本，$\mathbf{T} = \mathbf{XW}$ 表示完整的主成分分解，其中 $\mathbf{W}$ （$d \times k$）是矩阵，其列是 $\mathbf{X^TX}$ 的特征向量
- 为了降低输入的维数，我们只使用前 k 个特征向量，得到 $\mathbf{T_k} = \mathbf{XW_k}$（主成分分析降维后结果）
- 如果想要通过 $\mathbf{T_k}$ 重构 $\mathbf{X}$，可以有 $\mathbf{X'} = \mathbf{Y} = \mathbf{T_k}{\mathbf{W_k}}^T$







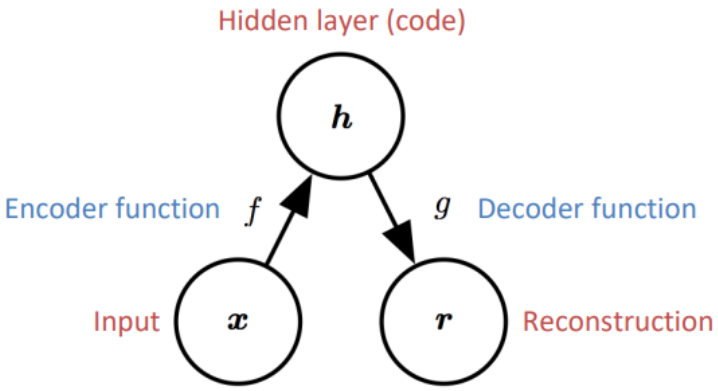
- 我们把 Original Data $\mathbf{X}_d$-> $\mathbf{T}_\mathbf{k}$ 称为编码
- 我们把 $\mathbf{T}_\mathbf{k}$ -> $\mathbf{Y}_d$ Reconstruction 称为解码
- 事实上，我们的目标就是想要找到一个能够压缩的结果 $\mathbf{T}_k$，使得它可以尽可能的还原原来的数据

PCA 的局限

- 我们捕捉原始空间中数据结构的能力受到我们考虑的转换类型的限制(正交线性)

3. 自动编码器 Autoencoders

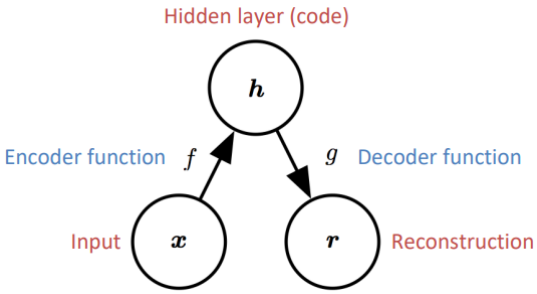
介绍



- 自动编码器是一种神经网络，它被训练成将输入复制到输出
- 它可以通过考虑非线性转换来推广主成分分析 PCA
- 使用 f 取学习一个特征
  - 特征的方差如何
  - 能否通过特征重构回输入
- 它与 PCA 的区别是，Encoder Function 和 Decoder Function 都是非线性的神经网络
- 我们希望强制编码具有比输入**更低的维度**
- 我们想要学习一种**低维表示**它能捕捉输入的**显著特征**

误差和反向传播

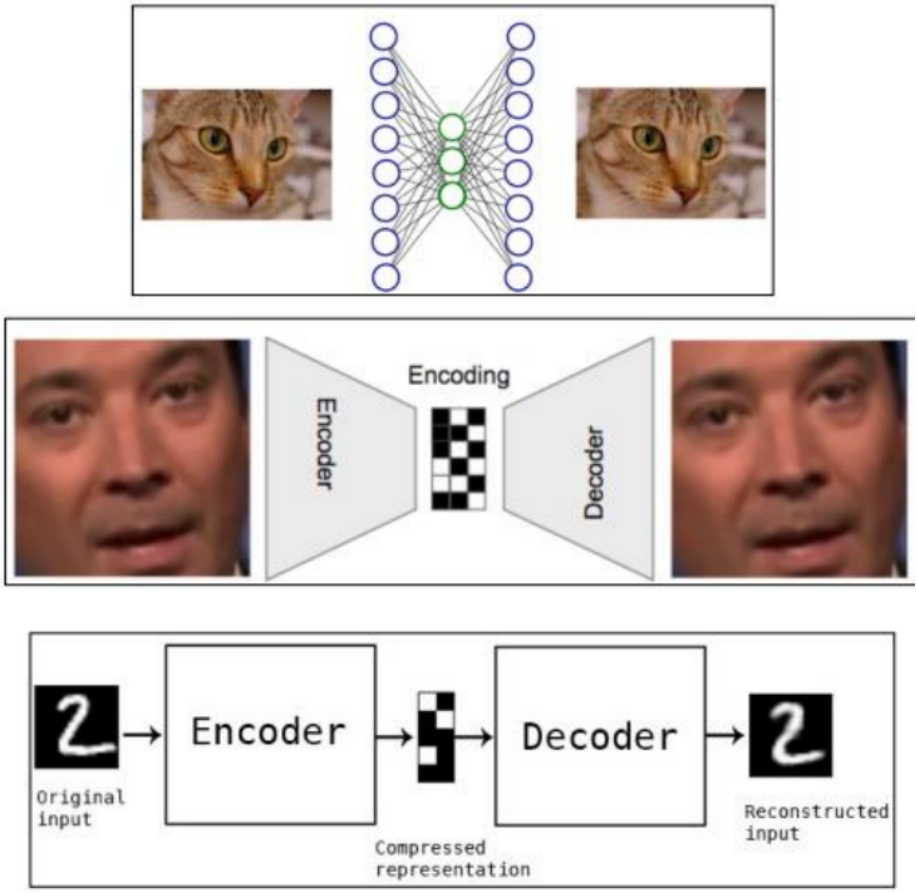
- 目标：最小化损失/重构误差



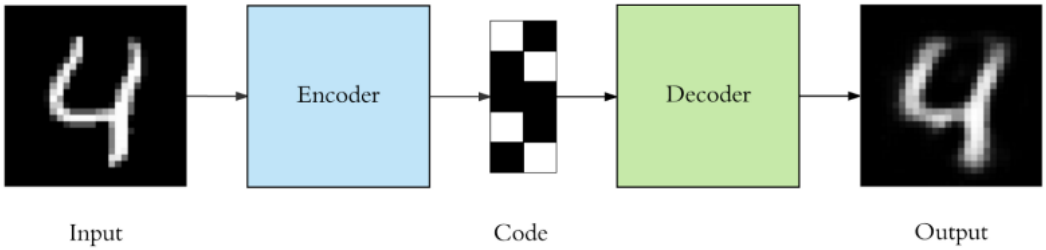
$$\frac{1}{N}\sum_{n=1}^{N} L\left(\boldsymbol{x}_n, \boldsymbol{r}_n\right) = \frac{1}{N}\sum_{n=1}^{N} L\left(\boldsymbol{x}_n, g\left(f\left(\boldsymbol{x}_n\right)\right)\right)$$

- 使用反向传播和 SGD 找出最优参数
- 如果使用 L2 范数去$L(x_n, r_n) = ||x_n - r_n||^2$，且 f 和 g 是线性的化，我们可以重构 PCA
- 当 x_n, r_n 的范围是 $[0, 1]$ 时，我们必须用交叉熵损失函数计算

自动编码器的架构



单层架构

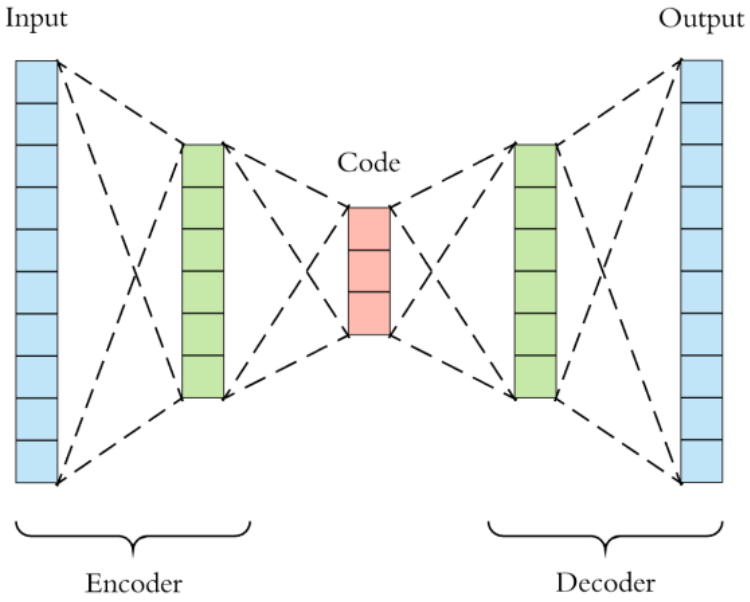


$$f(\boldsymbol{x}) = \mathrm{ReLU}(W\boldsymbol{x} + \boldsymbol{b}),\ \ g(\boldsymbol{x}) = \sigma(V f(\boldsymbol{x}) + \boldsymbol{c})$$

- 编解码器的神经网络结构很简单
  - W, V：权重层
  - b, c： bias

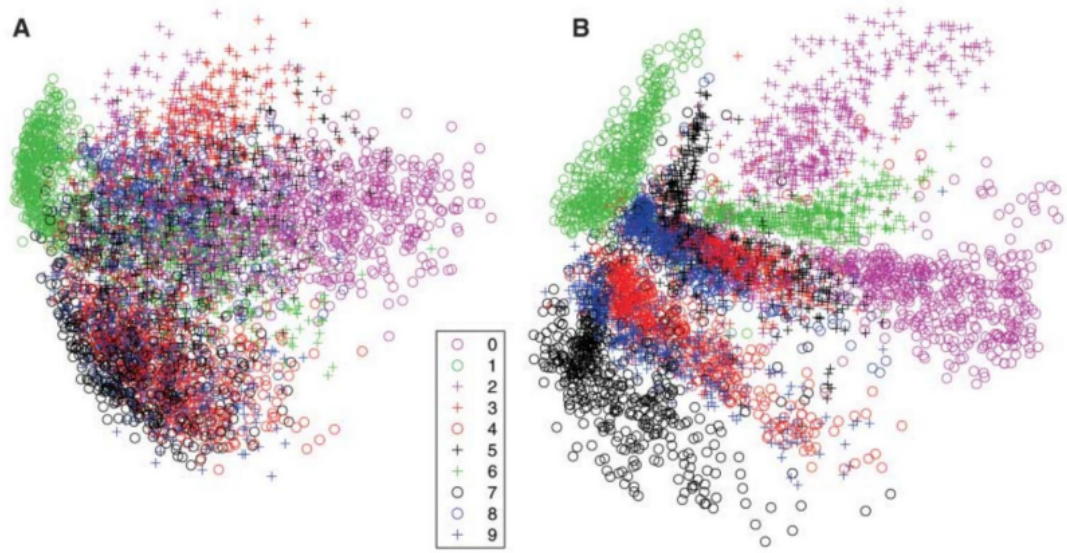
深度架构

在常规情况下，自动编码器的架构类似于下图



- 通常情况下
  - Encoder 的神经网络通常是 CNN
  - Decoder 的神经网络通常是上采样 + CNN

PCA 和 自动编码器

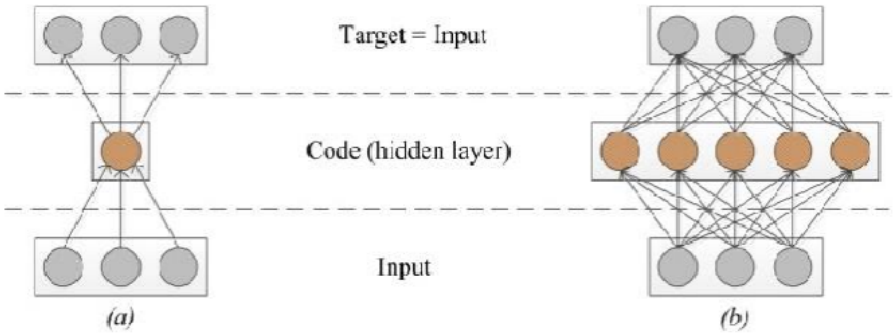


- 左图为 PCA 对于 1-9 数字降维到 2 维的效果
- 右图为自动编码器对于 1-9 数字降维到 2 维的效果

未完成的自动编码器 Undercomplete autoencoder

- 未完成的自动编码器
  - 编码的维度比输入的维度低
  - f 或者 g 的容量低
- 如果编码器和解码器有很大的容量我们会学习琐碎的变换（记忆）
  - 一个具有非常强大的非线性编码器的一维编码可以学习用编码 i 表示每个训练示例 x_i

过度完成的编码器 Overcomplete autoencoder



- 过度完成的编码器
  - h 的维度比 x 高
  - 必须被规范化

$$\frac{1}{N}\sum_{n=1}^{N} L\left(\boldsymbol{x}_n, g\left(f\left(\boldsymbol{x}_n\right)\right)\right) + \Omega(\boldsymbol{h})$$

- $\Omega(\boldsymbol{h})$：规范化
- 目标：除了能够将输入复制到输出之外，鼓励模型拥有其他有用的属性

规范化的编码器

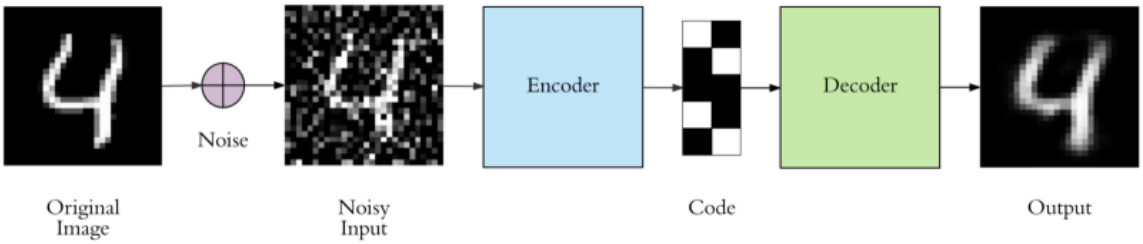
稀疏编码器 Sparse autoencoders

- 通过添加惩罚项来限制自动编码器的容量，以惩罚编码过大

$$\frac{1}{N}\sum_{n=1}^{N} L\left(\boldsymbol{x}_n, g\left(f\left(\boldsymbol{x}_n\right)\right)\right) + \lambda\|\boldsymbol{h}\|_1$$

- 通常用于提取鲁棒性强的特征或降低输入数据的维数

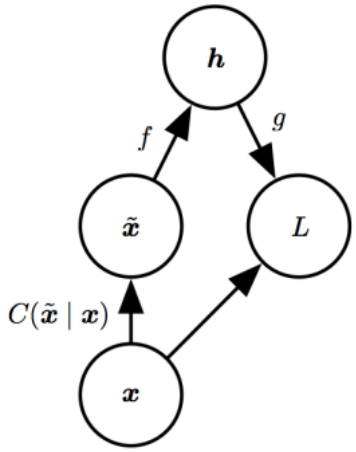
降噪编码器 Denoising autoencoders



- 不是增加惩罚，而是改变重构误差来解释噪声的引入

$$\frac{1}{N}\sum_{n=1}^{N} L\left(\boldsymbol{x}_n, g\left(f\left(\tilde{\boldsymbol{x}}_n\right)\right)\right)$$

- 输入的时候，将输入样本 x_i 添加噪声，构成 $\tilde{\boldsymbol{x}}_i$
- 噪声可以是**高斯噪声或 Dropout**，即部分输入被随机设置为 0
- 自动编码器学习降噪 map
- 网络被迫学习鲁棒性更高的表示
- 应用：图像去噪



- C：引入噪声的过程
- $\tilde{\boldsymbol{x}}$：带噪声的输入
- decoder 的输出和输入 $\mathbf{x}$ 进行比较

收缩自动编码器 Contractive autoencoder

- 训练自动编码器结合了两种强制需求
  - 重构：求出 $h = f(x)$，使 x 可以通过 $x = g(h)$ 求出
  - 有限容量：编码器不能表示任何可能的函数 f
- 这些强制需求共同推动隐藏的表示来捕获关于数据生成分布的结构的信息
- 自动编码器只能对重建训练数据所需的**变化**进行建模
- 如果训练数据集中在一个流形附近，只有在 x 附近与流形相切的变化需要对应 $h = f(x)$ 的变化
- 自动编码器学习如何捕捉流形的局部坐标系统

学习更明确的流形公式

$$\frac{1}{N}\sum_{n=1}^{N} L(\mathbf{x}_n, g(f(\mathbf{x_n}))) + \Omega(\mathbf{h}, \mathbf{x})$$

- 惩罚项：$\Omega(\mathbf{h}, \mathbf{x}) = \lambda||\frac{\partial f(\mathbf{x})}{\partial x}||_F^2$
  - 惩罚项是一个雅可比矩阵的 Frobenius 范数，即强迫编码器学习一个在训练样本周围没有太大变化的表示

自动编码器的应用

- 降维
- 图片降噪
- 特征提取
- 水印去除
- 图片上色

总结

- 自动编码器学习通过潜在空间对输入进行编码和解码
- 正则化是学习有用的表示法所必需的，它可以采取多种形式
- 这些表示可以用作特征或降低输入数据的维度
- 降噪和压缩自动编码器能够学习数据的流形结构