第十四章 自编码器

自编码器

- 神经网络的一种
- 一个由函数$\boldsymbol{h} = f(\boldsymbol{x})$表示的编码器
- 一个生成重构的解码器$\boldsymbol{r} = g(\boldsymbol{h})$
- 希望$\boldsymbol{h}$获得有用的特性，而不是简单的复制输入

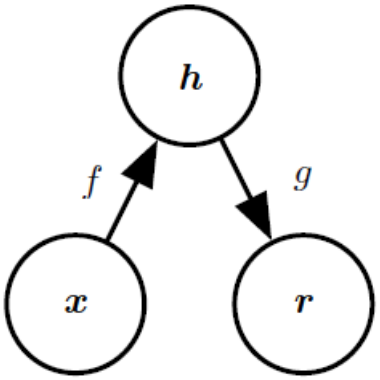


图 14.1: 自编码器的一般结构，通过内部表示或编码 $\boldsymbol{h}$ 将输入 $\boldsymbol{x}$ 映射到输出（称为重构）$\boldsymbol{r}$。自编码器具有两个组件：编码器 f（将 $\boldsymbol{x}$ 映射到 $\boldsymbol{h}$）和解码器 g（将 $\boldsymbol{h}$ 映射到 $\boldsymbol{r}$）。

随机编码器和解码器

在自编码器中，x既是输入也是目标

损失函数的具体形式视$p_{decoder}$的形式而定

将编码函数推广到编码分布，

- 潜变量模型：$p_{model}(\boldsymbol{h}, \boldsymbol{x})$
- 随机编码器：$p_{encoder}(\boldsymbol{h}|\boldsymbol{x}) = p_{model}(\boldsymbol{h}|\boldsymbol{x})$
- 随机解码器：$p_{decoder}(\boldsymbol{x}|\boldsymbol{h}) = p_{model}(\boldsymbol{x}|\boldsymbol{h})$

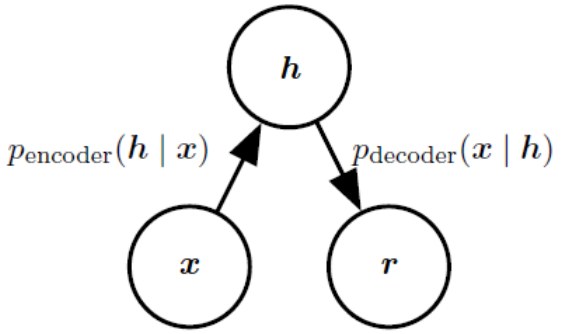


图 14.2: 随机自编码器的结构，其中编码器和解码器包括一些噪声注入，而不是简单的函数。这意味着可以将它们的输出视为来自分布的采样（对于编码器是 $p_{\text{encoder}}(\boldsymbol{h} \mid \boldsymbol{x})$，对于解码器是 $p_{\text{decoder}}(\boldsymbol{x} \mid \boldsymbol{h})$）。

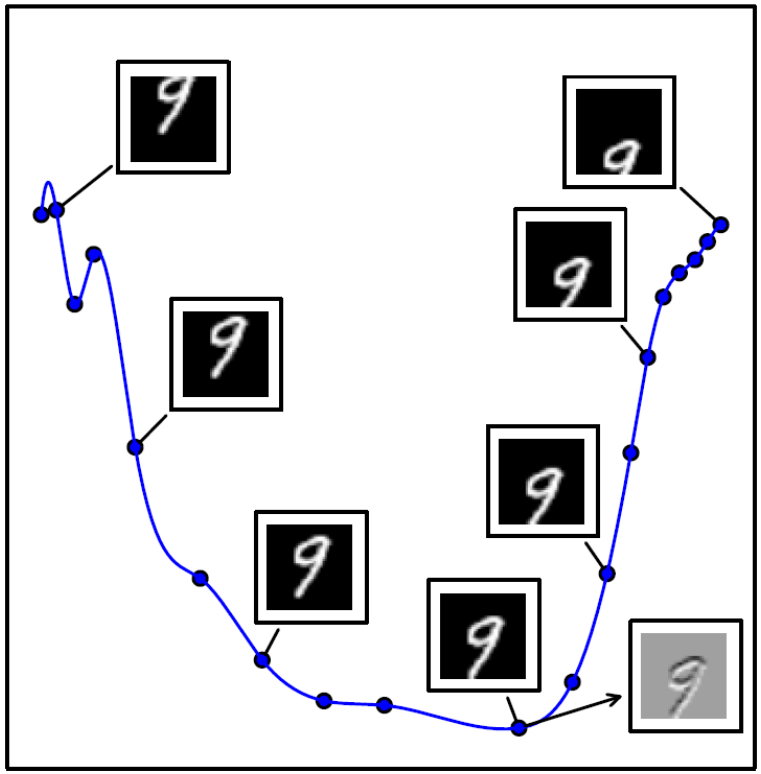
使用自编码器学习流形

所有自编码器的训练过程涉及两种推动力的折衷:

- 学习训练样本$\boldsymbol{x}$的表示$\boldsymbol{h}$使得$\boldsymbol{x}$能通过解码器近似地从$\boldsymbol{h}$中恢复
- 满足约束或正则惩罚

重要的原则:

- 自编码器必须有能力表示重构训练实例所需的变化



欠完备自编码器

限制$\boldsymbol{h}$的维度比$\boldsymbol{x}$小，

- 编码维度小于输入维度
- 强制自编码器捕捉训练数据中最显著的特征

- 当解码器是线性的且L是均方误差，与PCA相同
- 非线性编码器函数f和非线性解码器函数g的自编码器，能学习出PCA非线性推广

- 如果自编码器的容量太大，那训练来执行复制任务的自编码器可能无法学习到数据集的任何有用信息

正则自编码器

正则自编码器

- 不必限制模型容量
- 根据要建模的数据分布的复杂性，选择合适的编码维数和编码器、解码器容量

正则自编码器使用的损失函数可以鼓励模型学习其他特性

- 稀疏表示
- 表示的小导数
- 对噪声或输入缺失的鲁棒性

- 稀疏自编码器
- 去噪自编码器
- 收缩自编码器

稀疏自编码器

在训练时结合编码层的稀疏惩罚$\Omega(\boldsymbol{h})$和重构误差

$$L\left(x, g\left(f(x)\right)\right) + \Omega(\boldsymbol{h})$$

- 一般用来学习特征
- 正则项取决于数据

整个稀疏自编码器框架是对带有潜变量的生成模型的近似最大似然训练，而不将稀疏惩罚视为复制任务的正则化

- $-\log p_{\text{model}}(\boldsymbol{h}) = \sum_i (\lambda |h_i| - \log \frac{\lambda}{2}) = \Omega(\boldsymbol{h}) + \text{const}$
- 常数项只跟λ有关

去噪自编码器

接受损坏数据作为输入，预测原始未被损坏数据作为输出

- 从训练数据中采一个训练样本x
- 引入一个损坏过程$C(\tilde{\boldsymbol{x}}|\mathbf{x})$
- 从$C(\tilde{\boldsymbol{x}}|\mathbf{x})$采一个损坏样本$\tilde{\boldsymbol{x}}$
- 将$(\tilde{\boldsymbol{x}}, \mathbf{x})$作为训练样本来估计自编码器的重构分布 $p_{reconstruct}(\mathbf{x}|\tilde{\boldsymbol{x}}) = p_{decoder}(\boldsymbol{x}|\boldsymbol{h})$
- 对负对数似然$\log p_{decoder}(x|h)$进行基于梯度法的近似最小化

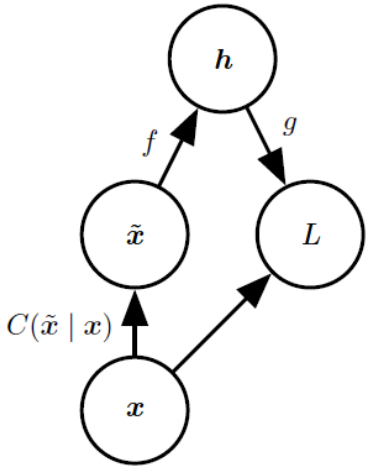


图 14.3: 去噪自编码器代价函数的计算图。去噪自编码器被训练为从损坏的版本 $\tilde{\boldsymbol{x}}$ 重构干净数据点 $\boldsymbol{x}$。这可以通过最小化损失 $L = -\log p_{\text{decoder}}(\boldsymbol{x} \mid \boldsymbol{h} = f(\tilde{\boldsymbol{x}}))$ 实现，其中 $\tilde{\boldsymbol{x}}$ 是样本 $\boldsymbol{x}$ 经过损坏过程 $C(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$ 后得到的损坏版本。通常，分布 p_{decoder} 是因子的分布（平均参数由前馈网络 g 给出）。

去噪自编码器

得分匹配最大似然的代替

- 提供了概率分布的一致估计，促使模型在各个数据点$\boldsymbol{x}$上获得与数据分布相同的得分
- 得分是一个特定的梯度场$\nabla_x \log p(x)$
- 学习$\log p_{data}$的梯度场是学习p_{data}结构的一种方式

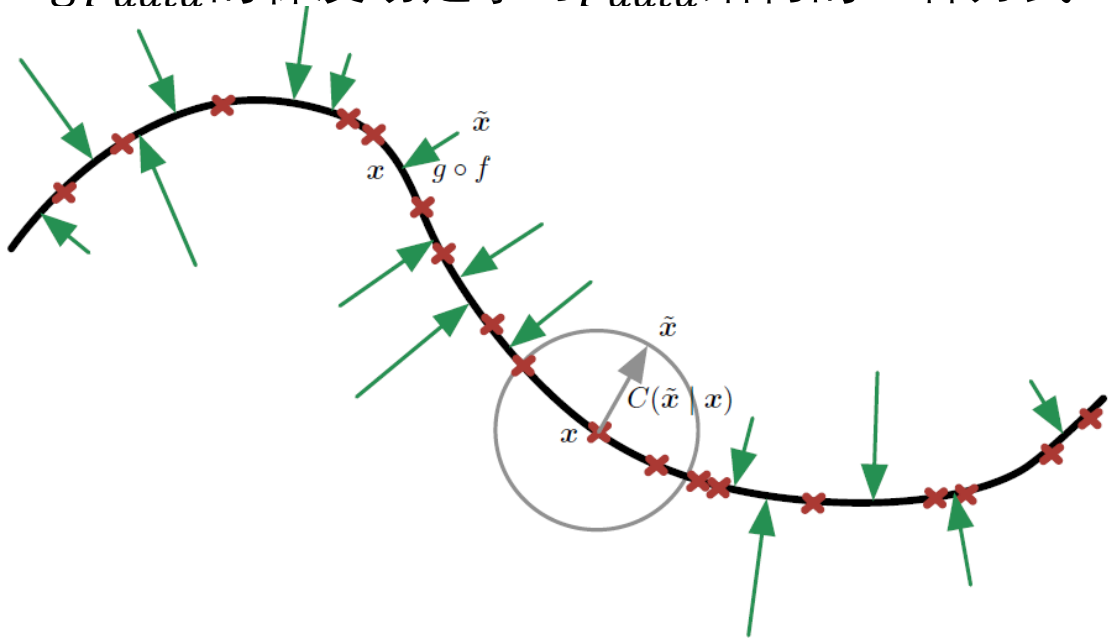


图 14.4: 去噪自编码器被训练为将损坏的数据点 $\tilde{\boldsymbol{x}}$ 映射回原始数据点 $\boldsymbol{x}$。我们将训练样本 $\boldsymbol{x}$ 表示为位于低维流形（粗黑线）附近的红叉。我们用灰色圆圈表示等概率的损坏过程 $C(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$。灰色箭头演示了如何将一个训练样本转换为经过此损坏过程的样本。当训练去噪自编码器最小化平方误

收缩自编码器

添加显示正则项:

- 鼓励f的导数尽可能小
- $\Omega(\boldsymbol{h}) = \lambda \left\| \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_F^2$

- 去噪自编码器能抵抗小且有限的输入扰动
- 收缩自编码器使特征提取函数能抵抗极小的输入扰动

问题:

- 尽管它在单一隐藏层的自编码器情况下是容易计算的，但在更深的自编码器情况下会变的难以计算
- 如果我们不对解码器强加一些约束，收缩惩罚可能导致无用的结果

预测稀疏分解

混合模型：

- 稀疏编码
- 参数自编码器

组成：

- 由一个编码器$f(\boldsymbol{x})$
- 一个解码器$g(\boldsymbol{h})$
- 优化过程：
  - 最小化：$\|\boldsymbol{x}-g(\boldsymbol{h})\|^2+\lambda|\boldsymbol{h}|_1+\gamma\|\boldsymbol{h}-f(\boldsymbol{x})\|^2$

- f是一个可微带参函数，计算相对简单
- PSD 模型可堆叠，并用于初始化其他训练准则的深度网络

自编码器的应用

降维:

- 低维表示可以提高许多任务的性能
- 小空间的模型消耗更少的内存和运行时间

信息检索:

- 从降维中获得一般益处，还使某些低维空间中的搜索变得极为高效
- 语义哈希
  - 通过降维和二值化的信息检索方法
  - 被用于文本输入