

日期： /

Autoregressive model

对于一个拟合 $p(x_1 \dots x_n)$ 的生成任务，有

$$P(x_1 \dots x_n) = P(x_1) P(x_2 | x_1) P(x_3 | x_1 x_2) \dots P(x_n | x_1 x_2 \dots x_{n-1})$$

假设 $P(x_1 \dots x_n) = P_{\text{CPT}}(x_1; \alpha^1) P_{\text{Logit}}(x_2 | x_1; \alpha^2) P_{\text{Logit}}(x_3 | x_1, x_2; \alpha^3) \dots$

$$P_{\text{Logit}}(x_n | x_1 \dots x_{n-1}; \alpha^n)$$

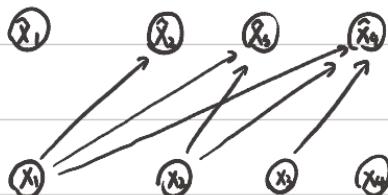
其中 $P_{\text{CPT}}(x_1=1; \alpha^1) = \alpha^1, P(x_1=0) = 1 - \alpha^1$

$$P_{\text{Logit}}(x_2=1 | x_1; \alpha^2) = \sigma(\alpha_0^2 + \alpha_1^2 x_1)$$

$$P_{\text{Logit}}(x_3=1 | x_1, x_2; \alpha^3) = \sigma(\alpha_0^3 + \alpha_1^3 x_1 + \alpha_2^3 x_2)$$

这就把生成问题拆解为J回归问题，这种策略称
为自回归。

[Fully Visible Sigmoid Belief Network / FVSBN]



条件变量 $x_i | x_1 \dots x_{i-1}$ 有参数 $\hat{x}_i = p(x_{i+1} | x_1 \dots x_{i-1}, \alpha^i)$
 $= p(x_{i+1} | x_{i+1}, \alpha^i)$ 的伯努利分布

日期： /

如何从自回归模型中采样？

关键是取得第一个先验概率 $\bar{X}_i \sim P(X_i)$, 之后的条件概率可以根据建立的模型取得

参数量：

在 $X_i \sim B(p)$ 的情况下，参数量为 $\sum_{i=1}^n i \approx \frac{n^2}{2}$

如何改进？

替换 logistic 回归，使用更深层的神经网络

[NAND] 即使用多层感知机替换 FSBN 后的结果。
 $\hat{X}_i = p(X_i | X_1, \dots, X_{i-1}; A_i, C_i, \alpha_i, b_i) = \sigma(\alpha_i h_i + b_i)$
 $h_i = \sigma(A_i X_{i-1} + C_i)$ [此处为 2 层 MLP]

通过权重绑定可以加速计算，例如：

$$h_1 = \sigma\left(\begin{pmatrix} w_1 & \vdots \\ \vdots & w_n \end{pmatrix} x_1 + c\right) \quad h_2 = \sigma\left(\begin{pmatrix} w_1 & w_2 & \vdots \\ \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + c\right) \quad h_3 = \sigma\left(\begin{pmatrix} w_1 & w_2 & w_3 & \vdots \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + c\right)$$

日期： /

对于 $h_i \in R^d$, 此时参数数量级为 n .

对于 X_i 可能取多元值的情况，令其满足

$$P(X_i | X_1, \dots, X_{i-1}) = \text{Cat}(p_1^i, \dots, p_k^i)$$

此时使用 Softmax 代替 logistic 函数

$$h_i = \sigma(W_{:, i} X_{\leq i} + b_i)$$

$$\hat{X}_i = (p_1^i, p_2^i, \dots, p_k^i) = \text{softmax}(A_i h_i + b_i)$$

对于连续取值的 X_i , 我们令 X_i 服从特定的连续概率分布, 例如混合 Gaussian 分布

$$P(X_i | X_1, \dots, X_{i-1}) = \sum_{j=1}^k \frac{1}{K} N(x_i; \mu_j^i, \sigma_j^i)$$

拟合方式为: $h_i = \sigma(W_{:, i} X_{\leq i} + b_i)$

$$\hat{X}_i = (\mu_1^i, \dots, \mu_k^i, \sigma_1^i, \dots, \sigma_k^i) = f(h_i)$$

[autoencoder/AE]

encoder (编码器): $e(x) = \sigma(W^T \sigma(W^T x + b^1) + b^2)$

decoder (解码器): $d(e(x)) \approx x$, $d(h) = \sigma(Vh + c)$

日期： /

使用重建损失：

离散： $w^l, w^b, b^l, b^b, v, c \min \sum_i \sum_j -x_i \log \hat{x}_i - (1-x_i) \log (1-\hat{x}_i)$

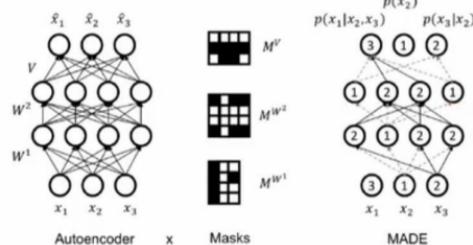
连续： $w^l, w^b, b^l, b^b, v, c \min \sum_i \sum_j (x_i - \hat{x}_i)^2$

- 通常情况下需限制 e 和 d 避免其退化为恒等函数
- AE 通常用于表示学习任务，而不能用于生成任务，这是因为 $e(x)$ 的分布是未知的
- AE 和自回归在表面上是相似的，但是 AE 缺少类似 FUSBN 或 NADE 内部的结构
- 使用自回归模型进行采样时：

- \hat{x}_i 不能依赖任何变量 $x \in (x_1, x_2, x_3)$
- \hat{x}_i 仅依赖 $x_{\leq i}$

如果要使用 AE 进行自回归，就必须依赖子掩码

[MADE]



日期： /

[RNN] $h_0 = b_0$

$$h_{t+1} = \tanh (W_{hh} h_t + W_{xh} X_{t+1})$$

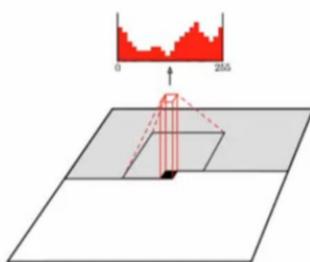
$$O_{t+1} = W_{hy} h_{t+1}$$

不足：训练慢，顺序依赖，顺序生成

当 X 含多个通道时，应当使用掩码机制

一种改进的方式是应用注意力机制

[Pixel CNN] 在图像问题中，使用 CNN 是自然的，确保模型的自回向结构也是关键所在



在卷积过程中，要对黑色的像素进行预测，使用灰色部分的信息是可以接受的，而白色部分的信息应当被掩蔽

自回向模型的问题是：① 无法提取特征

② 无法对数据点进行聚类

③ 无法进行无监督学习

日期： /

自回归模型使用极大似然的方式学习

[KL散度] 对于两个概率分布，KL散度定义为

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \quad \text{或} \quad D(p||q) = \int p(x) \log \frac{p(x)}{q(x)}$$

$D(p||q) \geq 0$ 且仅在 $p=q$ 时为 0

通常 q 为模型给出的概率分布， p 为实际概率分布

- KL散度是非对称的， $D(p||q) \neq D(q||p)$ 后者常用
于GAN中

- 最小化KL散度等价于最小化交叉熵损失

$$\underset{P_\theta}{\operatorname{argmin}} D(P_{\text{data}} || P_\theta) = \underset{P_\theta}{\operatorname{argmin}} -E_{x \sim P_{\text{data}}} [\log P_\theta(x)] = \underset{P_\theta}{\operatorname{argmax}} E_{x \sim P_{\text{data}}} [\log P_\theta(x)]$$

涉及 P_{data} 的先验分布。

$$\text{vs } E_D[\log P_\theta(x)] = \frac{1}{|D|} \sum_{x \in D} \log P_\theta(x) \text{ 评估} \xrightarrow{\text{无法直接优化}}$$

故优化目标转变为 $\max_{P_\theta} \frac{1}{|D|} \sum_{x \in D} \log P_\theta(x)$

即 $\max_{P_\theta} \prod_{x \in D} P_\theta(x)$ 即极大似然。

这是一个Monte Carlo估计。ws
采样估计真实分布

对于一个数据集，目标函数就是

$$L(\theta, D) = \prod_{j=1}^m P_\theta(x^{(j)}) = \prod_{j=1}^m \prod_{i=1}^{n_j} P_{\text{neural}}(x_i^{(j)} | x_{<i}^{(j)}, \theta_i)$$

日期： /

为了便于优化，令

$$l(\theta) = \log L(\theta, D) = \sum_{j=1}^m \sum_{i=1}^n \log P_{\text{neural}}(X_i^{(j)} | X_{<i}^{(j)}, \theta_j)$$

则 $\nabla_{\theta_j} l(\theta) = \sum_{j=1}^m \nabla_{\theta_j} \sum_{i=1}^n \log P_{\text{neural}}(X_i^{(j)} | X_{<i}^{(j)}, \theta_j)$

需对整个数据集求和，当 m 很大时成本过大

$$\begin{aligned}\nabla_{\theta_j} l(\theta) &= m \sum_{j=1}^m \underbrace{\frac{1}{m} \sum_{i=1}^n}_{\text{均匀分布的期望}} \nabla_{\theta_j} \log P_{\text{neural}}(X_i^{(j)} | X_{<i}^{(j)}, \theta_j) \\ &= m \underbrace{\mathbb{E}_{X^{(j)} \sim D} \left[\sum_{i=1}^n \nabla_{\theta_j} \log P_{\text{neural}}(X_i^{(j)} | X_{<i}^{(j)}, \theta_j) \right]}_{\text{(SGD的思想)}}\end{aligned}$$