PixelCNN

Tags

前置知识

自回归模型

AR模型的形式可以表示为：

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + \epsilon_t$$

其中：

x_t 是当前时刻的值，

$\phi_1, \phi_2, ..., \phi_p$ 是模型的参数（通过数据拟合得到）

p是模型的阶数，表示模型依赖于多少个过去的时间点

核心思想

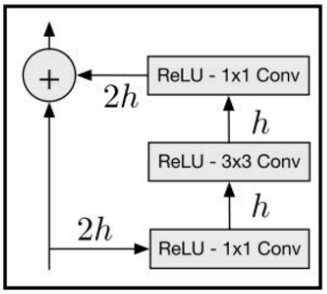
给图像的子像素定义一个先后顺序，之后让每个子像素的颜色取值分布由之前所有的子像素决定。通过条件化建模每个像素的值，使得图像的生成具有条件依赖关系。

PixelCNN 建立了条件概率模型：

$$P(\mathbf{X}) = \prod_{i=1}^{N} P(x_i | x_1, x_2, \ldots, x_{i-1})$$

模型架构

PixelCNN
7×7 conv mask A
Multiple residual blocks
3×3 conv mask B
ReLU followed by 1×1 conv, mask B (2 layers)
256-way Softmax for each RGB color (Natural images) or Sigmoid (MNIST)



residual block

注意两点：

1. 网络的输出是一个经softmax的概率分布。
2. 网络的所有卷积层要替换成带掩码的卷积层，第一个卷积层用A类掩码，后面的用B类掩码。

损失函数通常采用 **交叉熵损失**

$$L = -\sum_{i=1}^{N} \log P(x_i | x_1, \ldots, x_{i-1})$$

改进模型

Gated PixelCNN

消除了原PixelCNN里的视野盲区

PixelCNN++

1.使用logistic分布代替256路softmax

$$logistic(\mu, s) = \frac{1}{4s} sech^2(\frac{x-\mu}{2s})$$

$$v \sim \sum_{i=1}^{K} \pi_i logistic(\mu_i, s_i)$$

V为一个输出颜色

2.简化RGB子像素之间的约束关系

用简单的线性约束代替了原先较为复杂的用神经网络表示的子像素之间的约束

3.使用U-Net架构

4.使用dropout正则化

在每个子模块的第一个卷积后面加了一个Dropout