Deep Learning 2024011303 Liu Hanzuo

HW5

Q&A

P1

$$L(\theta) = \mathbb{E}_{x'}[\log \hat{p}(x)] = \mathbb{E}_{u\sim Uniform([0,1)^d), x\sim p_{data}}[\log \hat{p}(x+u)]$$
$$= \mathbb{E}_{x\sim p_{data}} \int_{[0,1)^d} \log \hat{p}(x+u)du \le \mathbb{E}_{x\sim p_{data}} \log \int_{[0,1)^d} \hat{p}(x+u)du$$
$$= \mathbb{E}_{x\sim p_{data}} \log p(x)$$

Note that the inequality mentioned here is from the Jenson inequality, whereas $(\log x)'' = -\frac{1}{x^2} < 0$, apply Jenson inequality and we get the inequality.

P2 We prove that the transformation T is invertible, while the Jacobian determinant is positive.

$$u_i = \frac{x_i - \mu_i(x_{<i})}{\exp(\sigma_i(x_{<i}))}$$

$$J_{i,j} = \frac{\partial x_i}{\partial u_j} = \begin{cases} \exp(\sigma_i(x_{<i})) & i = j \\ \frac{\partial \mu_i(x_{<i})}{\partial u_j} + u_i \cdot \frac{\partial \exp(\alpha_i(x_{<i}))}{\partial u_j} & i > j \\ 0 & i < j \end{cases}$$

thus

$$\det(J) = \prod_{i=1}^{L} \exp(\sigma_i(x_{<i})) > 0$$
$$p_X(x) = p_U(u) \cdot \det(J)^{-1}$$

Until here, we show that the Jacobian matrix is invertible and has its determinant is positive.

P3 1.

$$y = conv(m \cdot w, x)$$

Note that each layer we take a 3*3 layer as convolutional mask:

$$w = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Thus the visible mask for layer s is:

$$x[i-s:i+1, j:j+s+1] \cup x[i+l, j+l:j+s+1] (l=1,2,\cdots,s)$$

The prove could be showed by induction easily since that the transportation is either to left and above or has its $y_{delta} < x_{delta}$.

2. We divide the area into two parts: the horizontal part and the vertical part. Where the horizontal part is the rows above the current pixel, and the vertical part is the pixels at the same row and before the current pixel.

Horizontal Part:

$$w_A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, w_B = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

we update horizontal part individually by using the first layer's mask being w_A and w_B in each afterwards layer. Then in theory, the horizontal part could cover all areas above the current pixel.

Vertical Part:

$$k_A = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, k_B = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

we update the vertical part with two part: the horizontal and vertical (from last layer). For the vertical, we use k_A for the first layer and k_B for the second layer. The vertical part could cover all areas before the current pixel in the same row.

The update rule is:

$$H_i = (kernel_{h,i} \cdot w) * H_{i-1}$$

$$V_i = (kernel_{v,i} \cdot k) * V_{i-1} + H_i$$

Intersting Observation: I notice that we could view the H_i as the cell state and V_i has the hidden state in a LSTM, then the update rule that cell state only depends on itself and the hidden state depends on both cell state and hidden state is same to the LSTM, and I believe this is why that paper is called ”Recurrent PixelCNN”.