MathDNN HW 7

2018-13260 차재현

1 Problem 1

(a) WLOG assume that x obtains its maximum at index j. Write

$$\nu_\beta(x) = \frac{1}{\beta}\log\sum_i \exp(\beta x_i) = \frac{1}{\beta}\sum_i \exp(\beta(x_i - x_j)) + x_j,$$

where summation is over all $1 \le i \le n$. Since $x_j \ge x_i$ for all i, we have

$$0 \le \frac{1}{\beta}\sum_i \exp(\beta(x_i - x_j)) \le \frac{1}{\beta}\sum_i \exp(0) = \frac{n}{\beta},$$

therefore

$$x_j \le \nu_\beta \le x_j + \frac{n}{\beta}.$$

Taking $\beta \to \infty$ proves (a).

(b) Collecting the partial differential with each entry

$$\frac{\partial \nu_\beta}{\partial x_i} = \frac{\exp(\beta x_i)}{\sum_j \exp(\beta x_j)}$$

with $\beta = 1$ is

$$\nabla\nu_1 = \left(\frac{\exp(\beta x_1)}{\sum_j \exp(\beta x_j)}, \cdots, \frac{\exp(\beta x_n)}{\sum_j \exp(\beta x_j)}\right),$$

which is a softmax function.

(c) Denote i as unique maximal index. Uniqueness means $x_j - x_i < 0$ for all $j \ne i$. In this case, as $\beta \to \infty$, $\exp(\beta(x_j - x_i))$ converges to zero. Therefore

$$\lim_{\beta\to\infty} \frac{\exp(\beta x_j)}{\sum_k \exp(\beta x_k)} = \lim_{\beta\to\infty} \frac{\exp(\beta(x_j - x_i))}{\sum_k \exp(\beta(x_k - x_i))} = \frac{0}{1} = 0,$$

$$\lim_{\beta\to\infty} \frac{\exp(\beta x_i)}{\sum_k \exp(\beta x_k)} = \lim_{\beta\to\infty} \frac{\exp(\beta(x_i - x_i))}{\sum_k \exp(\beta(x_k - x_i))} = \frac{1}{1} = 1.$$

Collecting these results gives $\nabla\nu_\beta(x) \to e_i$ as $\beta \to \infty$.

2 Problem 2

First, consider convolutional layer.

We use the same method to problem 4 of Problem 5.

i) nn.Conv2d(3, 64, kernel_size=11, stride=4).

This converts input $(3, 27, 227)$ to $(64, 55, 55)$. The number of calculation is

$$64 \times 55 \times 55 \times (3 \times (2 \times 11^2 - 1) + 2 + 1) = 140,553,600.$$

After conv, maxpool converts shape to $(64, 27, 27)$.

ii) nn.Conv2d(64, 192, kernel_size=5, padding=2).

This converts input $(64, 27, 27)$ to $(192, 27, 27)$. The number of calculation is

$$192 \times 27 \times 27 \times (64 \times 2 \times 5^2) = 447,897,600.$$

After conv, maxpool converts shape to $(192, 13, 13)$.

iii) nn.Conv2d(192, 384, kernel_size=3, padding=1).

This converts input $(192, 13, 13)$ to $(384, 13, 13)$. The number of calculation is

$$384 \times 13 \times 13 \times (192 \times 2 \times 3^2) = 224,280,576.$$

iv) nn.Conv2d(384, 256, kernel_size=3, padding=1).

This converts input $(384, 13, 13)$ to $(256, 13, 13)$. The number of calculation is

$$256 \times 13 \times 13 \times (384 \times 2 \times 3^2) = 299,040,768.$$

v) nn.Conv2d(256, 256, kernel_size=3, padding=1).

This preserves input shape $(256, 13, 13)$. The number of calculation is

$$256 \times 13 \times 13 \times (256 \times 2 \times 3^2) = 199,369,512.$$

The total number of calculation occurs in convolutional layer becomes $1,311,142,056$.

Next, consider FC layer.

Consider we have input vector $v \in \mathbb{R}^m$ and output $w \in \mathbb{R}^n$ with bias. In this case, the total number of calculation becomes $n(2m - 1) + n = 2nm$.

i) nn.Linear(256 * 6 * 6, 4096).

The number of calculation is $2 \times (256 * 6 * 6) \times 4096 = 75,497,472$.

ii) nn.Linear(4096, 4096).

The number of calculation is $2 \times 4096 \times 4096 = 33,554,432$.

iii) nn.Linear(4096, 1000) (output shape is (1000,) in default). The number of calculation is $2 \times 4096 \times 1000 = 8,192,000$.

The total number of calculation occurs in linear layer becomes $117,243,904$. Therefore, during forward pass, addition and multiplication operate $1,428,385,960$ times.

3 Problem 4

(a) That $\partial y_L/\partial y_{L-1} = A_{w_L}$ and $\partial y_l/\partial y_{l-1} = \mathrm{diag}(\sigma'(A_{w_l}y_{l-1}+b_l\mathbf{1}_{n_l}))$ are proved in problem 6 of homework 4.

By definition, $(A_{w_l})_{ij} = w_l^{(j-i+1)}$ if $i \le j \le f_l - 1$, 0 otherwise. This gives

$$\frac{\partial y_L}{\partial w_l^{(k)}} = \frac{\partial y_L}{\partial y_l}\frac{\partial y_l}{\partial w_l^{(k)}} = \frac{\partial y_L}{\partial y_l}\mathrm{diag}(\sigma'(A_{w_l}y_{l-1} + b_l\mathbf{1}_{n_l}))(y_{l-1}^{(k)} + \cdots + y_{l-1}^{(n_{l-1}-f_l+k)})$$

$$= v_l(y_{l-1}^{(k)} + \cdots + y_{l-1}^{(n_{l-1}-f_l+k)}).$$

The last expression can be interpreted as follows: application 1D convolutional filter of size f_l with every entry v_l acts on y_{l-1}. Use $C_{v_l^\top}$ to express such convolutional filter. Combining over k, we have third equation

$$\frac{\partial y_L}{\partial w_l} = (C_{v_l^\top}y_{l-1})^\top.$$

Finally, by the same way as problem 6 of homework 4, we get

$$\frac{\partial y_L}{\partial b_l} = \frac{\partial y_L}{\partial y_l}\frac{\partial y_l}{\partial b_l} = \frac{\partial y_L}{\partial y_l}\mathrm{diag}(\sigma'(A_{w_l}y_{l-1} + b_l\mathbf{1}_{n_l}))\mathbf{1}_{n_l} = v_l\mathbf{1}_{n_l}.$$

(b) Denote $n = \mathrm{dim} v$, $v_a^b = (v_a, \cdots, v_{a+b-1})$, and $\odot$ = 1D convolution operation.

As in homework 1, describe matrix-vector operation $A_{w_i}v$ as

$$A_{w_i}v = (w_i \odot v_1^{f_i}, \cdots, w_i \odot v_{n-f_i}^{f_i})^\top.$$

vector-matrix operation is just transpose of above result, so we get

$$u^\top A_{w_i}^\top = (w_i \odot u_1^{f_i}, \cdots, w_i \odot u_{n-f_i}^{f_i}).$$