MathDNN HW 5

2018-13260 차재현

1 Problem 2

Recall gradients calculated in Problem 4:

$$\frac{\partial y_l}{\partial y_{l-1}} = \mathrm{diag}(\sigma'(A_l y_{l-1} + b_l))A_l = \mathrm{diag}(\sigma'(\tilde{y_{l-1}}))A_l,$$

$$\frac{\partial y_l}{\partial b_l} = \mathrm{diag}(\sigma'(A_l y_{l-1} + b_l)) = \mathrm{diag}(\sigma'(\tilde{y_{l-1}})),$$

$$\frac{\partial y_L}{\partial A_l} = \mathrm{diag}(\sigma'(A_l y_{l-1} + b_l)) \left(\frac{\partial y_L}{\partial y_l}\right)^\top y_{l-1}^\top = \mathrm{diag}(\sigma'(\tilde{y_{l-1}})) \left(\frac{\partial y_L}{\partial y_l}\right)^\top y_{l-1}^\top.$$

Consider the case when A_j is small. Let $i < j$ be chosen arbitrary. Derivative of composition of functions gives

$$\frac{\partial y_L}{\partial y_i} = \frac{\partial y_L}{\partial y_{L-1}} \frac{\partial y_{L-1}}{\partial y_{L-2}} \cdots \frac{\partial y_{i+1}}{\partial y_i}. \tag{1}$$

Since $i < j$, right side of (1) contains the term $\partial y_j / \partial y_{j-1} = \mathrm{diag}(\sigma'(A_j y_{j-1} + b_j))A_j$. Thus A_j appears, and (1) becomes small. In case of differentiation by b_i, we have

$$\frac{\partial y_L}{\partial y_i} = \frac{\partial y_L}{\partial y_{L-1}} \frac{\partial y_{L-1}}{\partial y_{L-2}} \cdots \frac{\partial y_{i+1}}{\partial y_i} \frac{\partial y_i}{\partial b_i}.$$

In case A_i, (1) already contained. Therefore both becomes small.

Next, assume the absolute value of $\tilde{y_j}$ is large. From the derivative of sigmoid

$$\sigma'(x) = \frac{e^{-x}}{(1+e^{-x})^2},$$

when absolute value of x becomes large, $\sigma'(x)$ becomes small. That $\tilde{y_j}$ has large absolute value means every entry of this vector has large absolute value, so $\sigma'(\tilde{y_j})$ becomes small. Using (1) and $\partial y_j / \partial y_{j-1} = \mathrm{diag}(\sigma'(A_j y_{j-1} + b_j))A_j$, $i < j$ implies derivative w.r.t both b_i and A_i has the term $\sigma'(\tilde{y_j})$. Thus both become small.

2 Problem 3

Initial case. Take $k = 0$, we get

$$\theta^1 - \theta^0 = \alpha g^0$$

at form I and

$$v^1 = g^0$$

$$\theta^1 - \theta^0 = -\alpha v^1 = -\alpha g^0$$

at form II.

Inductive case. Assume two forms are same at $k = n-1$. When $k = n$, form I becomes

$$\theta^{n+1} - \theta^n = -\alpha g^n + \beta(\theta^n - \theta^{n-1}).$$

Now, first equation of form II becomes, $v^{n+1} = g^n + \beta v^n$, together with induction hypothesis we get

$$\begin{aligned}\theta^{n+1} - \theta^n &= -\alpha v^{n+1} \\ &= -\alpha g^n + \beta(-\alpha v^n) \\ &= -\alpha g^n + \beta(\theta^n - \theta^{n-1}).\end{aligned}$$

Thus two results are same.

3 Problem 4

When input $X \in \mathbb{R}^{N\times a\times b}$ (n channel with $a \times b$ pixel) passes convolutinal filter $\{F_{n,m} \mid 1 \le n \le N, 1 \le m \le M\}$ with (kernel size, padding, channel) $= (2k+1, k, M)$ and activation σ, each entry $Y_{m,x,y}$ of $Y \in \mathbb{R}^{M\times a\times b}$ can be written as

$$Y_{m,x,y} = \sigma\left(\sum_n \sum_{-k\le i,j\le k} (F_{n,m})_{k+1+i,k+1+j} X_{n,x+i,j+j}\right),$$

where undefined region of X is 0 by padding. When $2|a, b$, and X passes maxpool with (kernel size, stride) = (2,2), $Y_{m,x,y}$ of $Y \in \mathbb{R}^{N\times a/2\times b/2}$ can be written as

$$Y_{m,x,y} = \max(X_{m,2x-i,2y-j} \text{ for } i,j \text{ in } \{0,1\}).$$

Applying these facts inductively.

1) Each $y_1[k,i,j]$ has its receptive field $\{X[c,m,n] \text{ defined} \mid |m-i|, |n-j| \le 1\}$.

2) Each $y_2[k,i,j]$ has its receptive field $\{y_1[c,m,n] \mid c = k, -1 \le m-2i, n-2j \le$

0}.
Together with 1), $\{X[c,m,n]$ defined $\mid -2 \leq m-2i, n-2j \leq 1\}$ affects $y_2[k,i,j]$.
3) This can be obtained by applying 2) twice.
We get $\{X[c,m,n]$ defined $\mid -4 \leq m-4i, n-4j \leq 2\}$ affects $y_3[k,i,j]$

4 Problem 5

(i) In homework 4, I point out that (a,b)-convolutional layer with k input channel has $abk+1$ parameters. Applying this fact, first network has $128(256 \cdot 1^2 + 1) + 192(256 \cdot 3^2 + 1) + 96(256 \cdot 5^2 + 1) = 1,089,952$ parameters, whereas second network has $128(256 \cdot 1^2 + 1) + (64(256 \cdot 1^2 + 1) + 192(64 \cdot 3^2 + 1)) + (64(256 \cdot 1^2 + 1) + 96(64 \cdot 5^2 + 1) + 64(256 \cdot 1^2 + 1) = 346,720$.

(ii) Consider a simplest case, a filter of size (a,b) acts once to produce 1 entry of output tensor.
First, elementwise multiplication occurs ab times. Next, ab elements are summed up by $ab-1$ times addition.
Next, suppose the number of input channel is n. This means first case repeats n times. After this we sum up n results and adding bias. This causes n additions. Lastly, activation applies. Total number of computation becomes following:
addition : $n(ab-1)+n = nab$, multiplication : nab, activation : 1.
Now we can compute the number of operation, by simply multiplying the number of output pixels to previous result. First, take a look on first network. The number of computation is

1. Upper 1×1 layer
    - addition : $32^2 \cdot 128 \cdot 256 = 33,554,432$
    - multiplication : $32^2 \cdot 128 \cdot 256 = 33,554,432$
    - activation : $32^2 \cdot 128 = 131,072$
2. Middle 3×3 layer
    - addition : $32^2 \cdot 192 \cdot 256 \cdot 3^2 = 452,984,832$

- multiplication : $32^2 \cdot 192 \cdot 256 \cdot 3^2 = 452,984,832$
- activation : $32^2 \cdot 192 = 196,608$

3. Lower 5×5 layer

- addition : $32^2 \cdot 96 \cdot 256 \cdot 5^2 = 629,145,600$
- multiplication : $32^2 \cdot 96 \cdot 256 \cdot 5^2 = 629,145,600$
- activation : $32^2 \cdot 96 = 98,304$

By the same way, that of second network is

1. Upper 1×1 layer

- addition : $32^2 \cdot 128 \cdot 256 = 33,554,432$
- multiplication : $32^2 \cdot 128 \cdot 256 = 33,554,432$
- activation : $32^2 \cdot 128 = 131,072$

2. Middle left 1×1 layer

- addition : $32^2 \cdot 64 \cdot 256 \cdot 1^2 = 16,771,216$
- multiplication : $32^2 \cdot 64 \cdot 256 \cdot 1^2 = 16,771,216$
- activation : $32^2 \cdot 64 = 65,536$

3. Middle right 3×3 layer

- addition : $32^2 \cdot 192 \cdot 64 \cdot 3^2 = 113,246,208$
- multiplication : $32^2 \cdot 192 \cdot 64 \cdot 3^2 = 113,246,208$
- activation : 196,608

4. Lower left 1×1 layer

- addition : $32^2 \cdot 64 \cdot 256 \cdot 1^2 = 16,771,216$
- multiplication : $32^2 \cdot 64 \cdot 256 \cdot 1^2 = 16,771,216$
- activation : $32^2 \cdot 64 = 65,536$

5. Lower right 1×1 layer

- addition : $32^2 \cdot 96 \cdot 64 \cdot 5^2 = 157,286,400$
- multiplication : $32^2 \cdot 96 \cdot 64 \cdot 5^2 = 157,286,400$
- activation : $32^2 \cdot 96 = 98,304$

6. Bottom 1×1 layer

- addition : $32^2 \cdot 64 \cdot 256 = 16,777,216$
- multiplication : $32^2 \cdot 64 \cdot 256 = 16,777,216$
- activation : $32^2 \cdot 64 = 65,536$

Although second model roughly constitutes of first model with three additional 1×1 convolutional layers, the number of parameters and operations(computations) are larger on first. This is due to the following reason. The number of parameters and computation grows rapidly when convolutional filter becomes larger, as we saw. More specifically, number of computation is proportional to the product of input channel and square of filter size. On the second network, we reduce the number of channel before "large" filter (i.e., 3×3 or 5×5) applies, which reduces input channel term. This is the role of "bottleneck".