MFDNN HW5

MinGyu Shin

April 5, 2024

Problem 1 : *Implementing backprop for MLP.*

Results of backpropagation by PyTorch's autograd and implementing myself are exactly same.

```
By PyTorch's autograd
dA:  tensor([[2.3943e-05, 3.7064e-05, 4.2687e-06, 6.3700e-06],
        [3.4104e-05, 5.2794e-05, 6.0804e-06, 9.0735e-06],
        [2.4438e-05, 3.7831e-05, 4.3571e-06, 6.5019e-06],
        [2.0187e-05, 3.1250e-05, 3.5991e-06, 5.3707e-06]])
db:  tensor([[4.8247e-05],
        [6.8722e-05],
        [4.9245e-05],
        [4.0678e-05]])
By implementing myself
dA: tensor([[2.3943e-05, 3.7064e-05, 4.2687e-06, 6.3700e-06],
        [3.4104e-05, 5.2794e-05, 6.0804e-06, 9.0735e-06],
        [2.4438e-05, 3.7831e-05, 4.3571e-06, 6.5019e-06],
        [2.0187e-05, 3.1250e-05, 3.5991e-06, 5.3707e-06]],
       grad_fn=<MmBackward0>)
db: tensor([[4.8247e-05, 6.8722e-05, 4.9245e-05, 4.0678e-05]],
       grad_fn=<MulBackward0>)
```

Figure 1: Result of P1

Problem 2 : *Vanishing gradients.*

Since

$$\frac{\partial y_L}{\partial b_i} = \frac{\partial y_L}{\partial y_{L-1}} \cdots \frac{\partial y_{i+1}}{\partial y_i} \frac{\partial y_i}{b_i} = A_L \mathrm{diag}(\sigma'(\tilde{y}_{L-1}))A_{L-1} \cdots \mathrm{diag}(\sigma'(\tilde{y}_i)),$$

$$\frac{\partial y_L}{\partial A_i} = \mathrm{diag}(\sigma'(\tilde{y}_i))(A_L \mathrm{diag}(\sigma'(\tilde{y}_{L-1}))A_{L-1} \cdots)^\intercal y_{i-1}^\intercal,$$

and,

$$\sigma' \leq 1/4,$$

if A_j is small, the LHS becomes small. Moreover

$$\sigma'(z) \to 0 \text{ as } z \to \pm\infty,$$

if $\tilde{y}_j$ has large absolute value, the LHS becomes small too.

Problem 3 : *Two forms of momentum SGD.*

For $n = 1$,

$$\theta_{\mathrm{I}}^1 = \theta^0 - \alpha g^0 + \beta(\theta^0 - \theta^{-1}) = \theta^0 - \alpha g^0$$

$$\theta_{\mathrm{II}}^1 = \theta^0 - \alpha v^1 = \theta^0 - \alpha(g^0 + \beta v^0) = \theta^0 - \alpha g^0$$

, thus Forms I and II produce same θ^1. Assume that for $n \in \mathbb{N}$, $\theta_{\mathrm{I}}^i = \theta_{\mathrm{II}}^i$ for $i = 1, \ldots, n$. Then we have

$$\begin{aligned}
\theta^n &= \theta^{n-1} - \alpha g^{n-1} + \beta(\theta^{n-1} - \theta^{n-2}) && (1)\\
&= \theta^{n-1} - \alpha(g^{n-1} + \beta v^{n-1}) && (2)\\
&\Rightarrow \theta^{n-1} - \theta^{n-2} = -\alpha v^{n-1}. && (3)
\end{aligned}$$

Then,

$$\begin{aligned}
\theta_{\mathrm{II}}^{n+1} &= \theta^n - \alpha g^n - \alpha\beta v^n \\
&= \theta^n - \alpha g^n - \alpha\beta(g^{n-1} + \beta v^{n-1}) \\
&= \theta^n - \alpha g^n - \alpha\beta g^{n-1} + \beta^2(\theta^{n-1} - \theta^{n-2}) \\
&= \theta^n - \alpha g^n - \alpha\beta g^{n-1} + \beta(\theta^n - \theta^{n-1} + \alpha g^{n-1}) \\
&= \theta^n - \alpha g^n + \beta(\theta^n - \theta^{n-1}) \\
&= \theta_{\mathrm{I}}^{n+1}.
\end{aligned}$$

Consequently by mathematical induction, Forms I and II produce same θ sequence.

Problem 4

For convolution layer with kernel size = 3, padding = 1,

$$r(y[k,i,j]) = X[:, (i-1:i+1), (j-1:j+1)]$$

where $r(y)$ is receptive field of y, : indicates all indices, and $(i:j)$ indicates from i to j indices. If index is out of range, round up to 1 or down to latest index. For maxpool layer with size 2×2,

$$r(y[k,i,j]) = X[k, (2i-1:2i), (2j-1:2j)].$$

Thus,

$$r(y_1[k,i,j]) = X[:, (i-2:i+2), (j-2:j+2)]$$

$$r(y_2[k,i,j]) = X[:, (2i-3:2i+2), (2j-3:2j+2)]$$

$$r(y_3[k,i,j]) = X[:, (4i-9:4i+6), (4j-9:4j+6)]$$

Problem 5 : *Non-CE loss function.*

Summary is below.

- Naïve inception module

  (i) The number of trainable parameters

  1) 1×1 convolution : $256 \times 128 + 128 = 32,896$

  2) 3×3 convolution : $256 \times 9 \times 192 + 192 = 442,560$

  3) 5×5 convolution : $256 \times 25 \times 96 + 96 = 614,496$

  $\Rightarrow$ Total $= 1,089,952$

  (ii) The number of additions

  1) 1×1 convolution : $256 \times 32 \times 32 \times 128 = 33,554,432$

  2) 3×3 convolution : $9 \times 256 \times 32 \times 32 \times 192 = 452,984,832$

  3) 5×5 convolution : $25 \times 256 \times 32 \times 32 \times 96 = 629,145,600$

  $\Rightarrow$ Total $= 1,115,684,864$

  (iii) The number of multiplications

  1) 1×1 convolution : $256 \times 32 \times 32 \times 128 = 33,554,432$

  2) 3×3 convolution : $9 \times 256 \times 32 \times 32 \times 192 = 452,984,832$

  3) 5×5 convolution : $25 \times 256 \times 32 \times 32 \times 96 = 629,145,600$

  $\Rightarrow$ Total $= 1,115,684,864$

  (iv) The number of activation function evaluations

  1) 1×1 convolution : $32 \times 32 \times 128 = 131,072$

  2) 3×3 convolution : $32 \times 32 \times 192 = 196,608$

  3) 5×5 convolution : $32 \times 32 \times 96 = 98,304$

  $\Rightarrow$ Total $= 425,984$

- Inception module with 1×1 bottleneck convolutions

  (i) The number of trainable parameters

  1) 1×1 convolution : $256 \times 128 + 128 = 32,896$

  2) 1×1 convolution before 3×3 convolution : $256 \times 64 + 64 = 16,448$

  3) 3×3 convolution : $64 \times 3 \times 3 \times 192 + 192 = 110,784$

  4) 1×1 convolution before 5×5 convolution : $256 \times 64 + 64 = 16,448$

  5) 5×5 convolution : $64 \times 5 \times 5 \times 96 + 96 = 153,696$

  6) 1×1 convolution after maxpool : $256 \times 64 + 64 = 16,448$

  $\Rightarrow$ Total $= 346,720$

(ii) The number of additions

1) 1×1 convolution : $256 \times 32 \times 32 \times 128 = 33,554,432$

2) 1×1 convolution before 3×3 convolution : $256 \times 32 \times 32 \times 64 = 16,777,216$

3) 3×3 convolution : $64 \times 9 \times 32 \times 32 \times 192 = 113,246,208$

4) 1×1 convolution before 5×5 convolution : $256 \times 32 \times 32 \times 64 = 16,777,216$

5) 5×5 convolution : $64 \times 25 \times 32 \times 32 \times 96 = 157,286,400$

6) 1×1 convolution after maxpool : $256 \times 32 \times 32 \times 64 = 16,777,216$

$\Rightarrow$ Total $= 354,418,688$

(iii) The number of multiplications

1) 1×1 convolution : $256 \times 32 \times 32 \times 128 = 33,554,432$

2) 1×1 convolution before 3×3 convolution : $256 \times 32 \times 32 \times 64 = 16,777,216$

3) 3×3 convolution : $64 \times 9 \times 32 \times 32 \times 192 = 113,246,208$

4) 1×1 convolution before 5×5 convolution : $256 \times 32 \times 32 \times 64 = 16,777,216$

5) 5×5 convolution : $64 \times 25 \times 32 \times 32 \times 96 = 157,286,400$

6) 1×1 convolution after maxpool : $256 \times 32 \times 32 \times 64 = 16,777,216$

$\Rightarrow$ Total $= 354,418,688$

(iv) The number of activation function evaluations

1) 1×1 convolution : $32 \times 32 \times 128 = 131,072$

2) 1×1 convolution before 3×3 convolution : $32 \times 32 \times 64 = 65,536$

3) 3×3 convolution : $32 \times 32 \times 192 = 196,608$

4) 1×1 convolution before 5×5 convolution : $32 \times 32 \times 64 = 65,536$

5) 5×5 convolution : $32 \times 32 \times 96 = 98,304$

6) 1×1 convolution after maxpool : $32 \times 32 \times 64 = 65,536$

$\Rightarrow$ Total $= 622,592$

SUMMARY

(i) The number of trainable parameters : $1,089,952 \gg 346,720$

(ii) The number of additions : $1,115,684,864 \gg 354,418,688$

(iii) The number of multiplications : $1,115,684,864 \gg 354,418,688$

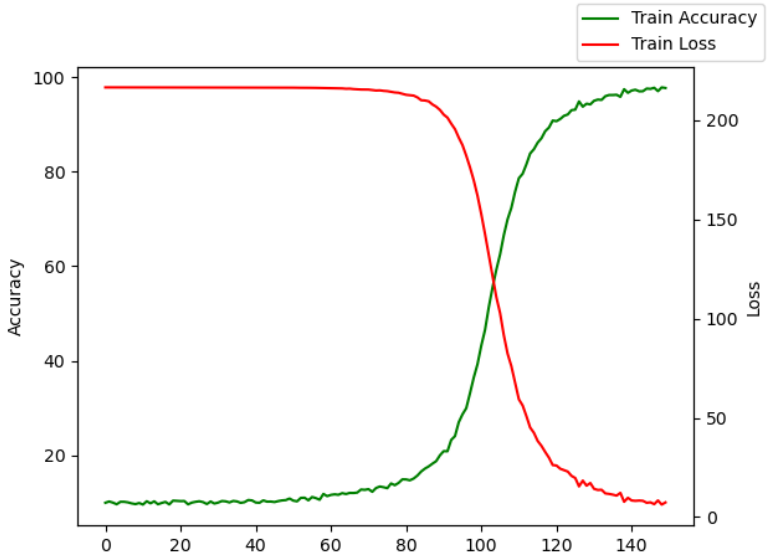
(iv) The number of activation function evaluations : $425,984 < 622,592$

Problem 6 : *Large neural networks memorize and interpolate training data.*

The result of the experiment considering MNIST data with completely randomized labels on a variation of AlexNet architecture is below.

```
Epoch 150 / 150
Train Loss: 7.3942
Train accuracy: 97.68%
Total training time: 823.3526818752289
```

(a) Train Loss and Train Accuracy of the last epoch



(b) Train Loss and Train Accuracy throughout the entire epoch

Since the used architecture may sufficiently large, the model trained the randomized label almost completely.