Name: Yuxuan Zhang

University of Illinois Spring 2020

CS 446/ECE 449 Machine Learning

Homework 5: Deep Net

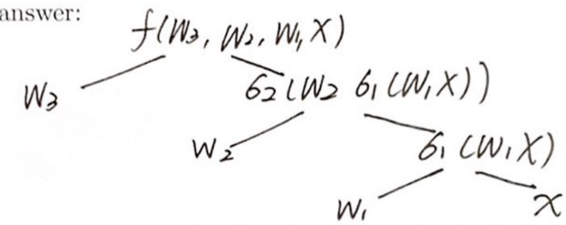
Due on Thursday March 5 2020, noon Central Time

1. **[25 points]** Deep Net

We want to train a simple deep net $f(w_3, w_2, w_1, x) = w_3\sigma_2(\underbrace{w_2\sigma_1(\underbrace{w_1 x}_{x_1})}_{x_2})$ where $w_1, w_2, w_3, x \in \mathbb{R}$ are real valued and $\sigma_1(x) = \sigma_2(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function.

(a) (1 point) Draw the computation graph that is specified by the function f.

Your answer:



(b) (2 points) Compute $\frac{\partial \sigma_1}{\partial x}$ and provide the answer (1) using only x, the exp-function and the square function, and (2) using only $\sigma_1(x)$.

Your answer:

(1) . $\frac{\partial 6_1}{\partial X} = \frac{+\exp(-X)}{(1+\exp(-X))^2}$ $\exp(-X)+1 = \frac{1}{6_1(X)}$

$\exp(-X) = \frac{1}{6_1(X)} - 1$

(2) $\frac{\partial 6_1}{\partial X} = 6_1^2(X)\left(\frac{1}{6_1(X)} - 1\right) = 6_1(X) - 6_1^2(X)$

(c) (2 points) Describe briefly what is meant by a 'forward pass' and a 'backward pass'?

Your answer:

Forward Pass: We pass the result from inner function to outer funiction. In deep learning, it is a forward pass to the next layer.

Backward Pass: When computing the ~~derati~~ derivatives, of a outer function of inner ~~φ~~ weights, partial derivatives of functions will be repeatedly used. Therefore in deep neural networks, it needs ~~backward pass~~ to pass derivatives backward.

Name: Yinxuan Zhang

(d) (2 points) Compute $\frac{\partial f}{\partial w_3}$. Which result should we retain from the forward pass in order to make computation of this derivative easy?

Your answer:

$$\frac{\partial f}{\partial W_3} = \sigma_2(W_2 \sigma_1(W_1 x))$$

We should retain $\sigma_1(W_1 x)$ and $\sigma_2(W_2 \sigma_1(W_1 x))$ from the forward pass.

(e) (3 points) Compute $\frac{\partial f}{\partial w_2}$. Make use of the second option obtained in part (b). Which results should we retain from the forward pass in order to make computation of this derivative easy?

Your answer:

$$\frac{\partial f}{\partial W_2} = \frac{\partial(W_3 \cdot \sigma_2(W_2 \cdot \sigma_1(W_1 x)))}{\partial W_2} = W_3 \frac{\partial(\sigma_2(W_2 \sigma_1(W_1 x)))}{\partial W_2} = W_3 \frac{\partial \sigma_2}{\partial (W_2 \sigma_1)} \sigma_1(W_1 x)$$

$$= W_3 \frac{\partial \sigma_2}{\partial x_2} \sigma_1(W_1 x) \quad \text{let } x_2 = W_2 \sigma_1(W_1 x)$$

$$= W_3 \sigma_1(W_1 x)[\sigma_2(x_2) - \sigma_2^2(x_2)]$$

We should retain $\sigma_1(W_1 x)$, ~~and~~ $\cancel{\frac{\partial \sigma_2}{\partial x_2}}$ $x_2 = W_2 \sigma_1(W_1 x)$.

(f) (5 points) Compute $\frac{\partial f}{\partial w_1}$. Make use of the second option obtained in part (b). Which results should we retain from the forward pass in order to make computation of this derivative easy? In what order should we compute the derivatives $\frac{\partial f}{\partial w_3}$, $\frac{\partial f}{\partial w_2}$ and $\frac{\partial f}{\partial w_1}$ in order to obtain the result as early as possible and in order to reuse as many results as possible. How is this order related to the forward pass?

Your answer:

let $x_2 = W_2 \sigma_1(W_1 x)$

$$\frac{\partial f}{\partial W_1} = W_3 \frac{\partial \sigma_2(W_2 \sigma_1(W_1 x))}{\partial W_1} = W_3 \frac{\partial \sigma_2}{\partial x_2} \cdot \frac{\partial x_2}{\partial W_1} = W_3 \frac{\partial \sigma_2}{\partial x_2} \cdot W_2 \frac{\partial(\sigma_1(W_1 x))}{\partial W_1}$$

$$= W_3 \frac{\partial \sigma_2}{\partial x_2} \cdot W_2 \cdot \frac{\partial \sigma_1}{\partial x_1} \cdot x \quad \text{let } x_1 = \sigma_1(W_1 x)$$

$$= W_3 W_2 \cdot x \cdot \frac{\partial \sigma_2}{\partial x_2} \cdot \frac{\partial \sigma_1}{\partial x_1} = W_3 W_2 \cdot x (\sigma_1 - \sigma_1^2)(\sigma_2 - \sigma_2^2).$$

We need to retain. x, $\sigma_1(W_1 x)$. $\sigma_2(W_2 \sigma_1(W_1 x))$.

We need to compute. first. $\frac{\partial f}{\partial W_3}$, then $\frac{\partial f}{\partial W_2}$. last $\frac{\partial f}{\partial W_1}$

It is the opposite order from forward pass.

Name: Yuxuan Zhang

(g) (2 points) We now want to train a convolutional neural net for 10-class classification of MNIST images which are of size 28×28. As a first layer we use 20 2d convolutions each with a filter size of 5×5, a stride of 1 and a padding of 0. What is the output dimension after this layer? Subsequently we apply max-pooling with a size of 2×2. What is the output dimension after this layer?

Your answer: $28-5+1=24$. ①. $24 \times 24 \times 20$.

②. $12 \times 12 \times 20$.

(h) (4 points) After having applied the two layers (convolution + pooling) designed in part (g) we want to use a second convolution + max-pooling operation. The max-pooling operation has a filter size of 2×2. The desired output should have 50 channels and should be of size 4×4. What is the filter size, the stride, the padding and the channel dimension of the second convolution operation?

Your answer:
Filter Size: ~~5x5~~ 5x5 Stride 1. Padding 0 Channel dim: 50.

$4 \times 4 \xLeftarrow{\text{max-pooling}} 8 \times 8 \xLeftarrow{12-5+1} 12 \times 12$

(i) (4 points) Complete `A5_DeepNet.py` by first implementing the two operations, where each operation is convolution+pooling. We also want to apply two linear layers, which you must implement. The first one maps from a $50 \cdot 4 \cdot 4$ dimensional space to a 500 dimensional one. After each convolution and after the first linear layer, we also want to apply ReLU non-linearities. Provide the entire deep net class here. What is the best test set accuracy that you observed during training with this architecture? How many parameters does your network have (including biases)?

Your answer:

```
def __init__(self):
    super(Net, self).__init__()
    self.conv1 = nn.Conv2d(1, 20, 5)
    self.conv2 = nn.Conv2d(20, 50, 5)
    self.pool = nn.MaxPool2d(2, 2)
    self.fc1 = nn.Linear(50*4*4, 500)
    self.fc2 = nn.Linear(500, 10)

def forward(self, x):
    x = self.pool(F.relu(self.conv1(x)))
    x = self.pool(F.relu(self.conv2(x)))
    x = view(-1, 50*4*4)
    x = F.relu(self.fc1(x))
    return self.fc2(x)
```

The best accuracy I get is 99.2.

Total parameters. 431080