Name:

University of Illinois Spring 2020

CS 446/ECE 449 Machine Learning
Homework 5: Deep Net

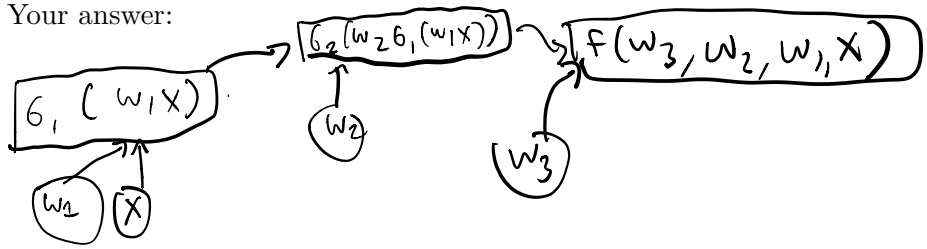
Due on Thursday March 5 2020, noon Central Time

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

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

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

   Your answer:

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

   Your answer:

   $$\sigma_1(u) = \frac{1}{1+\exp(-u)} = (1+\exp(-u))^{-1}$$

   $$\frac{\partial \sigma_1(u)}{\partial u} = -(1+\exp(-u))^{-2} \cdot [\exp(-u)(-1)]$$

   $$= \frac{\exp(-u)}{[1+\exp(-u)]^2}$$

   $$= \frac{1}{1+\exp(-u)} \cdot \frac{\exp(-u)}{1+\exp(-u)}$$

   $$= \sigma_1(u) \cdot (1 - \sigma_1(u))$$

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

   Your answer:

   Forward pass refers to sending labeled data through our neural in the forward direction, ie passing in the parameters (the weights and the input features) at the leaf nodes and running them through the computation graph to get a final result at the top most node.

   Backward pass is the how we actually update the weight parameters. When we get the result out of the computation graph, we can compare it to the label for that particular data point using some sort of cost function. In the case of regression, we can use mean squared error and in the case of classification, we can use cross entropy. This loss function is essentially just a giant function of all of our parameters and tells us how good our model predicted that data point. We take the derivative of this cost with respect to all the parameters and update them for each data point in our training set.

Name:

(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:

$\frac{\partial f}{\partial w_3} = \sigma_2(x_2)$.

$\sigma_2(x_2)$ should be retained

(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:

$$\frac{\partial f}{\partial w_2} = \frac{\partial f}{\partial \sigma_2} \cdot \frac{\partial \sigma_2}{\partial x_2} \cdot \frac{\partial x_2}{\partial w_2}$$

$$= w_3 \cdot \left[\sigma_2(x_2) \cdot (1 - \sigma_2(x_2))\right] \cdot \sigma_1(x_1)$$

$\sigma_2(x_2)$ & $\sigma_1(x_1)$ should be retained

(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:

$$\frac{\partial f}{\partial w_1} = \frac{\partial f}{\partial \sigma_2} \cdot \frac{\partial \sigma_2}{\partial x_2} \cdot \frac{\partial x_2}{\partial \sigma_1} \cdot \frac{\partial \sigma_1}{\partial x_1} \cdot \frac{\partial x_1}{\partial w_1}$$

$$w_3 \cdot \left[\sigma_2(x_2)(1-\sigma_2(x_2))\right] \cdot w_2 \cdot \left[\sigma_1(x_1)(1-\sigma_1(x_1))\right] \cdot x$$

We should retain $\sigma_1(x_1)$ & $\sigma_2(x_2)$ from forward pass.

There is not an optimal order to calculate the derivatives for this example. As long as you calculate and save the partial derivative df/dx2, which I highlight in pink in part e and f, whenever we calculate df/dw1 or df/dw2. As this partial df/dx2 shows up in both, we can use it to save computation. Generally, we should calculate the partials in the opposite direction of forward pass, in this case calculating df/dw3, followed by df/dw2 (saving the partial df/dx2) followed by df/dw1. For deeper nets, this order ensures that we can reuse as many of the calculations as possible.

Name:

(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:

after convolution

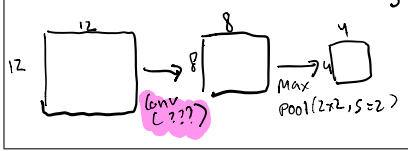
$O = \frac{(28 - 5 + 2(0))}{1} + 1 = 24 \times 24 \times 20$

after pooling

$O = 12 \times 12 \times 20$.

(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, and the channel dimension of the second convolution operation, assuming that padding is 0?

Your answer:



$8 = \frac{(12 - K + 2(0))}{S} + 1$

Stride = 1

filter size = 5×5

If we ↑ stride, filter size becomes negative

Channel In = 20

Channel out = 50

(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 your entire code pertaining to the "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:

```
49
50 class Net(nn.Module):
51     def __init__(self):
52         super(Net, self).__init__()
53         ################################
54         ## declare the layers of the network which have parameters
55         ################################
56         self.conv1 = torch.nn.Conv2d(1, 20, kernel_size=5, stride=1, padding=0)
57         self.conv2 = torch.nn.Conv2d(20, 50, kernel_size=5, stride=1, padding=0)
58         self.fc1 = torch.nn.Linear(50 * 4 * 4, 500)
59         self.fc2 = torch.nn.Linear(500, 10)
60
61     def forward(self, x):
62         ################################
63         ## combine the layers; don't forget the relu and pooling operations
64         ################################
65         x = F.relu(self.conv1(x))
66         x = F.max_pool2d(x, 2, 2)
67         x = F.relu(self.conv2(x))
68         x = F.max_pool2d(x, 2, 2)
69         x = x.view(-1, 50 * 4 * 4)
70         x = self.fc1(x)
71
72         return self.fc2(x)
73
```

431080 parameters

Best test set accuracy: 99.15%