Prof. Dr. Andreas Hotho, Albin Zehe, Jonas Kaiser
Data Science Chair

12./13.06.2024

6. Assignment in "Machine Learning for Natural Language Processing"

Summer Term 2024

1 General Questions

1. What is the effect of pooling layers (max/average pooling) in Convolutional Neural Networks? How often are they commonly used?
2. How are convolutions interpreted in NLP tasks?

? Something to think about

3. Can we implement a fully connected layer in a neural network by using a convolutional layer? If so, how?

   Is the opposite possible, i.e. can we imitate the behaviour of a convolutional layer using a fully connected layer?

2 Backpropagation in Convolutional Neural Networks

Given the matrix

$$M = \begin{pmatrix} 1 & 1 & 1 & 2 & 3 \\ 1 & 1 & 1 & 2 & 3 \\ 1 & 1 & 1 & 2 & 3 \\ 2 & 2 & 2 & 2 & 3 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \end{pmatrix}$$

and a kernel

$$k = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix},$$

assuming $L = \sum_{row=1}^{4} \sum_{col=1}^{3} O_{row,col}$ and $O = M * k$, calculate $\frac{\delta L}{\delta O}$, $\frac{\delta L}{\delta k}$, $\frac{\delta L}{\delta M}$. You do not need to apply the individual equations. For clarity, in this case " $*$ " refers to the actual convolution and not just cross-correlation.

3 Python

3.1 Implementing Neural Networks Part 4 — Optimisers

In this assignment, you will implement a neural network "library" yourself, using `Python` and `Numpy` (**`import numpy as np`**). The tool is inspired by PyTorch's implementation. This week, you will adapt the optimisers from the first assignment to train your neural network.

1. In the first assignment, you have already implemented some common optimisers for neural networks (SGD, Nesterov Momentum and Adam).

   In this exercise, you will add optimizer functions for SGD, simple Momentum, and Adam to our framework. Add classes `SGD`, `Momentum`, and `Adam` to your code. Their constructor should get the instantiated `NeuralNetwork` object and the necessary hyper-parameters such as the learning rate. The classes should provide a method called `update` that gets the weight and bias gradients from the gradient calculation. This method takes one step of the corresponding optimizer and updated the weights and biases of the given neural network model.

   ```
   class SGD:
       def __init__(self, nn:object, lr:float):
           #...
           pass

       def update(self, W_grads: List[np.array], b_grads:
        ↪  List[np.array]) -> None:
           #...
           pass
   ```

```python
class Momentum:
    def __init__(self, nn: object, lr: float, mu:
    ↪ float):
        #...
        pass

    def update(self, W_grads: List[np.array], b_grads:
    ↪ List[np.array]) -> None:
        #...
        pass


class Adam:
    def __init__(self, nn: object, lr: float, beta1:
    ↪ float, beta2: float):
        #...
        pass

    def update(self, W_grads: List[np.array], b_grads:
    ↪ List[np.array]) -> None:
        #...
        pass
```

2. Train the neural network you have implemented so far using these optimisers! For example, you can use the network to learn the XOR-Function:

x	y	XOR(x,y)
0	0	0
0	1	1
1	0	1
1	1	0