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

05./06.06.2024

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

Summer Term 2024

1 General Questions

1. How can we apply a CNN model to text?
   a) What is the input to the network?
   b) How do we work with different text lengths in one batch and across batches?
   c) What are the sizes of the CNN kernels?

> ? Something to think about
> Could we use **mean**-over-time pooling in the TextCNN by Kim? What problem can arise in situations where we have texts of different length?

2 Neural Network Hiccups

Dying ReLUs

A frequently used activation function for neural networks is the ReLU-function:

$$ReLU(x) = \begin{cases} x, & x > 0 \\ 0, & \text{else} \end{cases}$$

ReLUs sometimes suffer from the so-called "dying ReLU" problem. In this assignment, you will see what this means and how it can occur.

Assume a neural network with a scalar output, that is, the final layer of the network consists of only one neuron $o = \sum_{i=1}^{n} w_i h_i$, where w are the weights of the layer and h

is the output of the previous layer. Let's put a ReLU activation after that layer, to get $y = ReLU(o)$.

We use squared error as the loss function of the network:

$$L = se(y, t) = \frac{1}{2}(t - y)^2,$$

where t is the true label for the input.

Let $w = \begin{pmatrix} 0.2 & -0.3 & 0.5 \end{pmatrix}$, $h = \begin{pmatrix} 0.1 & 0.5 & 10.0 \end{pmatrix}$ and $t = 0.7$.

Perform the following steps:

1. Compute the gradient $\frac{\partial L}{\partial w}$!
2. Update the weights w using gradient descent with a learning rate of $\lambda = 0.1$
3. Repeat steps (1) and (2) with the updated weights, the input $h = \begin{pmatrix} 0.3 & 0.1 & 1.0 \end{pmatrix}$ and $t = 0.1$.
4. Repeat steps (1) and (2) with the updated weights, the input $h = \begin{pmatrix} 0.5 & 0.25 & 5 \end{pmatrix}$ and $t = 1$.

Describe what you find!

3 Convolutions

3.1 Manual Convolution

Given the matrix

$$M = \begin{pmatrix} 0 & 6 & 1 & 1 & 6 \\ 7 & 9 & 3 & 7 & 7 \\ 3 & 5 & 3 & 8 & 3 \\ 8 & 6 & 8 & 0 & 2 \\ 4 & 3 & 2 & 9 & 1 \end{pmatrix}$$

and a kernel

$$k = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Calculate the convolution $C = M * k$ using zero padding to make the output matrix C have the same size as M and a stride of 1! Is there a difference between a convolution and a cross correlation in this case?

3.2 Rotated Convolution

When deriving the formula for backpropagation over a convolutional layer, a substitution of a form similar to this is done:

$$\begin{aligned}\frac{\partial L}{\partial x_{a,b}} &= \sum_{m=-l'}^{l'} \sum_{n=-l'}^{l'} \delta_{h_{a+m,b+n}} w_{-m+l'+1,-n+l'+1} \\ &= \delta_{h_{a-l':a+l',b-l':b+l'}} * rot_{180}(w)\end{aligned}$$

where x is the input matrix, $w \in \mathbb{R}^{r \times r}$ is the kernel matrix and $l' := \left\lfloor \frac{r}{2} \right\rfloor$ is half the width/height of the kernel matrix.

Show that the double sum on the left-hand side can indeed be expressed by the convolution with rotated kernel on the right-hand side! It is not necessary to mathematically prove this relation. There are other ways, such as tracking the transformation of single indices.

4 Python

4.1 Implementing Neural Networks Part 3 — Dropout

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 implement Dropout regularisation.

For this, implement a new module (`forward` and `backward` function) that is initialised with a parameter p, denoting the probability that a weight is dropped. Do not forget to scale the resulting weights to make up for the missing weights.

```
class Dropout:
    def __init__(self, p=0.5):
        self.p = p

    def forward(self, x: np.array) -> np.array:
        #...
```

```
def backward(self, grad: np.array = np.array([[1]]))
 ↪  -> np.array:
    #...
```

Modify the implementation of the `NeuralNetwork` to include dropout with a specified rate (`p:float=0.5` in the constructor) after every hidden layer! Then check that each run of the program will result in different results due to the random cancellation of weights.