Technische Universität Hamburg
Institut für Mathematik
Lehrstuhl Numerische Mathematik

Jens-Peter M. Zemke
Jonas Grams

Mathematics of Neural Networks
winter semester 2021/2022
exercise sheet 7

Exercise 1: (2 points) Use the extended Euclidean algorithm to compute a GCD g of the two polynomials

$$a(u) := u^4 + 3u^3 - 8u^2 - 12u + 16, \quad b(u) := u^4 + u^3 - 7u^2 - u + 6,$$

and two polynomials c, d, such that

$$ac + bd = g$$

holds true.
Hint: You can implement the extended Euclidean algorithm (See Algorithm 3.1 on page 122 of the lecture notes[1]) with the Python package `sympy`. Use `sympy.quo(f, g)` to obtain the quotient of polynomials `f` and `g`. The computed output has rational coefficients and thus has to be scaled.
Use the following skeleton:

```python
"""
quo computes for given f,g the quotient q with
f = q*g + r for some rest term r

degree returns the degree of a given polynomial

expand returns a polynomial in standard representation
expand(f) = \sum_{i = 0}^n f_i x^i
"""
from sympy     import quo, degree, expand
from sympy.abc import u

def extended_euclidean_algorithm(a, b):

  """
  TODO Implement the extended Euclidean algorithm
       with sympys div
  """
  pass

if __name__ == '__main__':

  a = u**4 + 3*u**3 - 8*u**2 - 12*u + 16
  b = u**4 + u**3 - 7*u**2 - u + 6

  r, c, d = extended_euclidean_algorithm(a, b)
```

[1] version from 01.11.2021

```
    print('GCD =', r)
    print('c =', c)
    print('d =', d)
    print('Error in Bezout =', expand(a*c + b*d - r))
```

Exercise 2: (3+1 points)

a) Compute a minimal Winograd algorithm for the full convolution $\mathbf{z} = \mathbf{x} * \mathbf{f}$, where $\mathbf{x} \in \mathbb{R}^3$ and $\mathbf{f} \in \mathbb{R}^2$. Use the four points $g_1 = 0$, $g_2 = 1$, $g_3 = -1$, and $g_4 = \infty$.

b) Give the matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ of the matrix form

$$\mathbf{z} = \mathbf{A}^{\mathsf{T}}((\mathbf{B}\mathbf{x}) \circ (\mathbf{C}\mathbf{f})).$$

Hint: In the lecture you developed a minimal algorithm for $\mathbf{x} \in \mathbb{R}^2$ and $\mathbf{f} \in \mathbb{R}^2$ (see page 124 of the lecture notes[1]). Extend this example to $\mathbf{x} \in \mathbb{R}^3$ and $\mathbf{f} \in \mathbb{R}^2$.

Exercise 3: (2+1+2* points)

a) Let $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{f} \in \mathbb{R}^k$, and $\mathbf{y} \in \mathbb{R}^{n+k-1}$ be given. Prove that

$$\sum_{i=0}^{n-1} x_i \sum_{j=0}^{k-1} y_{i+j} f_j = \sum_{p=0}^{n+k-2} y_p \sum_{\ell=0}^{p} x_\ell f_{p-\ell},$$

where undefined entries x_ℓ and $f_{p-\ell}$ are zero.

b) Show that the full convolution of $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{f} \in \mathbb{R}^k$ is related to the computation of the valid cross-correlation of $\mathbf{y} \in \mathbb{R}^{n+k-1}$ and $\mathbf{f} \in \mathbb{R}^k$ or FIR filter $F(n,k)$

$$\begin{pmatrix} y_0 & y_1 & \cdots & y_{k-1} \\ y_1 & y_2 & \cdots & y_k \\ \vdots & \vdots & & \vdots \\ y_{n-1} & y_n & \cdots & y_{n+k-2} \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{k-1} \end{pmatrix}$$

by computing the scalar product of the resulting vector with $\mathbf{x} \in \mathbb{R}^n$ and using part a).

c) Use this result to derive a minimal algorithm for $F(3,2)$ from a minimal algorithm for the full convolution of $\mathbf{x} \in \mathbb{R}^3$ and $\mathbf{f} \in \mathbb{R}^2$.

Hint: A change of the summation order may be helpful for part c)

Exercise 4: (3 points) Compare the computation time for the different implementations of the evaluation of a convolutional layer. In addition to the already implemented methods `evaluate_scipy()`, `evaluate_fft()` and `evaluate_im2col()` you can find an implementation of the Winograd approach for filters $F \in \mathbb{R}^{3,3}$ called `evaluate_winograd()` in `layers.py`. Compare them using a different number of images, corresponding height and width and channels for randomly generated input. For example you can try

- 100 and 20 for the number of inputs,
- 3 and 50 for the number of channels,
- 28 and 100 for the height/width of the input.

What can you oberserve?
You can use the following skeleton.

```python
from time   import process_time
from layers import *

"""
TODO Test the evaluations for different numbers of inputs,
     channels, and different input dimensions
"""
n, c, h, w = None

m, fh, fw = 32, 3, 3
tensor = (c, h, w)
fshape = (m, fh, fw)
conv = Conv2DLayer(tensor, fshape)

# generate random input
x = np.random.randint(0, 10, (n, c, h, w))

# test winograd
print('---------------winograd----------------')
start = process_time()
"""
TODO Add the evaluation of the convolutional layer with
     the wingrad minimal algorithm
"""
time_winograd = process_time() - start
print('Time used by winograd:', time_winograd, 'seconds')

# test im2col
print('---------------im2col----------------')
start = process_time()
"""
TODO Add the evaluation of the convolutional layer with
     the im2col method
"""
time_im2col = process_time() - start
print('Time used by im2col:', time_im2col, 'seconds')

# test
print('---------------fft----------------')
start = process_time()
"""
TODO Add the evaluation of the convolutional layer with
     the the fft method
"""
time_fft = process_time() - start
print('Time used by fft:', time_fft, 'seconds')

# test
print('---------------scipy----------------')
start = process_time()
"""
TODO Add the evaluation of the convolutional layer with
     the scipy convolutions
"""
time_scipy = process_time() - start
print('Time used by scipy:', time_scipy, 'seconds')
```