TECHNICAL UNIVERSITY OF DENMARK

ADVANCED MACHINE LEARNING. MODULE 3.

EXERCISE 3. THEORY AND GENERATIVE MODELS

CONTENTS

Exercise A Graph convolutions

A discrete-time signal is given as a sequence of values $x[n]$, where n is an integer time index. The convolution between two discrete signals $x[n]$ and $h[n]$ is defined as

$$y[n] = (x \star h)[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

The convolution is a linear operation performed on two discrete-time signals to produce a third signal. It is a fundamental operation in signal processing and is used for various tasks such as filtering: Here, $x[n]$ is the input signal, $h[n]$ represents the filter (it is called the impulse response), and $y[n] = (x \star h)[n]$ is the output signal. Depending on $h[n]$, the filtering operation can e.g. amplify or attenuate different frequency components in the signal.

Question A.1: Consider the following input signal

$$x[n] = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases}$$

which is known as an *impulse*. What is the resulting output of the filter?

The impulse response of the filter $h[n]$ tells us how the system responds across time to a signal applied at time $n = 0$. We usually have $h[n] = 0 \;\forall\; n < 0$ so that the system does not respond before the input signal is applied. This is called a *causal* filter.

Let us define the unit delay operator D,

$$Dx[n] = x[n-1].$$

When we apply the operator D to a signal, it will be shifted (delayed) one unit in time.

Question A.2: Show that the convolution of a signal with a causal filter can be written as

$$\begin{aligned} y[n] =& h[0]x[n] + h[1]Dx[n] + h[2]D^2x[n] + \dots \\ =& \left(\sum_{k=0}^{\infty} h[k]D^k\right) x[n] \end{aligned}$$

Now, we will consider graph convolutions: A signal $\boldsymbol{x}[n]$ is defined on the nodes of a graph where each component of $\boldsymbol{x}[n]$ is a time varying scalar node signal that propagates through the graph from nodes to their neighbors. In the case of a discrete time signal, we saw how the unit delay operator D shifts the signal in time. On the graph, the signal at each nodes spreads to its neighbors at each time step, which can be written mathematically as a matrix multiplication with the adjacency matrix followed by a time delay. Thus, we have the following *neighborhood shift* operator

$$D\boldsymbol{x}[n] = \boldsymbol{A}\boldsymbol{x}[n-1]$$

With this operator, we can write the graph convolution of a signal with a causal filter as

$$\boldsymbol{y}[n] = (\boldsymbol{x} \star_{\mathcal{G}} h)[n] = \left(\sum_{k=0}^{\infty} h[k]D^k\right) \boldsymbol{x}[n]$$

This equation describes how the signal spreads in space (across the graph) and in time.

Question A.3: Assume that the filter $h[n]$ is only nonzero for $0 \le n \le N$, and apply a constant signal $\boldsymbol{x}[n] = \boldsymbol{x}$. What is the value of the signal $\boldsymbol{y}[n]$? (Hint: It is a constant independent of n.) Compare your result with Eq. 7.21 in the book.

Exercise B Graph Fourier transform

The discrete Fourier transform (DFT) is a mathematical operation that converts a finite length sequence (signal) $x[n]$ into a sequence of complex numbers $\tilde{x}[k]$, representing the signal's frequency content. The DFT is widely used in signal processing for analyzing and manipulating signals in the frequency domain. Given a sequence $x[n]$ of length N, the DFT is defined as

$$\tilde{x}[k] = \sum_{n=0}^{N-1} x[n] e^{-i\frac{2\pi k}{N}n}.$$

The DFT essentially computes the inner product of the input sequence $x[n]$ with a set of complex exponential functions, each representing a different frequency component. The resulting complex numbers $\tilde{x}[k]$ represent the amplitude and phase of each frequency component in the input signal.

Now, let us consider a *cycle graph* with N vertices; a simple undirected graph arranged in a circular manner, where each vertex is connected to its two adjacent vertices. Formally we can define it by the vertex set $\mathcal{V} = \{0, 1, \ldots, N-1\}$ and the edge set $\mathcal{E} = \{(0,1), (1,2), (2,3), \ldots, (N-2, N-1), (N-1, 0)\}$ such that each pair of consecutive vertices are connected, and the last vertex is connected to the first vertex, forming a closed loop. For example, for $N = 6$ the adjacency matrix is given by

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Consider a vector $\boldsymbol{u}_k$ with components

$$(\boldsymbol{u}_k)_n = e^{-i\frac{2\pi k}{N}n}$$

This is a complex exponential, i.e. a sinusoidal where the integer k is the normalized frequency.

Question B.1: Show that $\boldsymbol{u}_k$ is an eigenvector of the adjacency matrix of the cycle graph, $\boldsymbol{A}\boldsymbol{u}_k = \lambda \boldsymbol{u}_k$. Hint: One way to proceed is to show that the relation holds for each component of the eigenvalue problem vector, i.e. that $(\boldsymbol{A}\boldsymbol{u}_k)_n = \lambda_k (\boldsymbol{u}_k)_n$ for some λ_k.

Since the complex exponential $\boldsymbol{u}_k$ is an eigenvector of the adjacency matrix for any integer frequency k, we have demonstrated that the eigenvalue decomposition of the adjacency matrix of the cycle graph yields the DFT basis. We can generalize this to define the graph Fourier basis for any graph as the eigenvectors of the adjacency matrix.

Given the eigendecomposition

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^*$$

we can compute the graph Fourier transform (GFT) of a signal $\boldsymbol{x}$ as $\tilde{\boldsymbol{x}} = \boldsymbol{U}^*\boldsymbol{x}$ and the inverse GFT as $\boldsymbol{x} = \boldsymbol{U}\tilde{\boldsymbol{x}}$.

Question B.2: Show that the graph convolution of a signal with a causal filter can be written in the spectral domain as

$$\tilde{\boldsymbol{y}} = \sum_{k=0}^{\infty} h[k] \boldsymbol{\Lambda}^k \tilde{\boldsymbol{x}}$$

Thus we now have two ways to compute the graph convolution with a finite-length filter: Directly in the vertex domain, and in the spectral domain:

$$\boldsymbol{y} = \sum_{k=0}^{N} h[k] \boldsymbol{A}^k \boldsymbol{x} = \boldsymbol{U} \left(\sum_{k=0}^{N} h[k] \boldsymbol{\Lambda}^k \right) \boldsymbol{U}^* \boldsymbol{x} \tag{1}$$

Question B.3: Is there an advantage of the spectral approach in terms of computational complexity? Consider the necessary operations and their computational complexity (as they scale with the graph size). Consider what might be pre-computed when learning the filter $h[n]$.

Exercise C Programming exercise

In this exercise you will work with a graph convolution model for graph-level classifcation implemented in the script `graph_convolution.py`.

Like last week, we will use the *MUTAG dataset* introduced by Debnath et al.: a collection of nitroaromatic compounds (molecular graphs) and the task is to predict their mutagenicity on Salmonella typhimurium (graph-level binary classification). Vertices represent atoms and edges represent bonds, and the 7 discrete node labels represent the atom type (one-hot encoded). There are a total of 188 graphs in the dataset.

Question C.1: Examine and run the script.

- Go through each code block to remind yourself about the structure of the script. The script is very similar to last week's script, except for the model definition.

Question C.2: Examine the model definition in `SimpleGraphConv`, and go through the details in the `__init__` and `forward` functions.

- Notice how the graph filter is defined and initialized: It is initialized so that $h[0] = 1$ and $h[k]$ for $k > 1$ are small random numbers. Think about what a graph convolution does when only $h[0] = 1$ and the remaining coefficients are zero. Why might this be a reasonable initialization?
- Make sure you understand the role and function of `to_dense_adj` and `to_dense_batch`. Look up their documentation. How do they handle the fact, that not all graphs have the same number of nodes?
- Examine the for-loop that computes the graph convolution. Match it up against the formulas in the theoretical exercises and the book.
- Notice how the output filter is defined. What is its role and dimensions?

Question C.3: Modify the implementation of the graph convolution so that it is computed in the spectral domain, rather than in the vertex domain. The current implementation is based on the formula:

$$\boldsymbol{y} = \sum_{k=0}^{N} h[k]\boldsymbol{A}^k\boldsymbol{x}$$

and your task is to change the implementation to use this formula:

$$\boldsymbol{y} = \boldsymbol{U}\left(\sum_{k=0}^{N} h[k]\boldsymbol{\Lambda}^k\right)\boldsymbol{U}^*\boldsymbol{x}$$

Hint: You need to compute the eigenvalue decomposition. I suggest using the function `torch.linalg.eigh` which ensure that the result is real (not complex). This makes the implementation a bit easier.

- The two implementations should give identical results (except perhaps for small numerical differences). Test your implementation against the original.

Question C.4: (Optional) Modify the implementation so that the eigenvalue decompositions are precomputed and given as input to the `forward` function. Additionally, you can also precompute the dense matrices from `to_dense_batch` and `to_dense_adj`. This should significantly speed up the training.