Chapter 4: Real-world data representation using tensors

Main Points

- Neural networks require data to be represented as multidimensional numerical tensors (often 32-bit floating point)
- PyTorch often expects data to be organized into specific dimensions according to the model architecture
- The PyTorch tensor API can help to reshape the data as required

Subchapter summaries

4.1: Working with images

- An image is represented as a collection of scalars arranged in a regular grid with a height and a width (in pixels)
    - Single scalar per grid point (ie. pixel) --> grayscale
    - Multiple scalars per grid point --> difference colours or depths
- PyTorch modules dealing with image data require tensors to be laid out as C x H x W: channels, height, and width
- A dataset of multiple images then is N x C x H x W where N is the batch size
- Two suggested options for normalizing the data:
    - Divide the values of the pixels by 255 (the maximum representable number for 8-bit unsigned integers)
    - Compute the mean and standard deviation of the input data and scale it so that the output has zero mean and unit standard deviation across each channel

4.2: 3D images: Volumetric data

- 3D medical images like CT scans only have a single intensity channel, similar to a grayscale image, so raw data typically has three dimensions similar to 4.1
- A dataset of 3D images is represented by a 5D tensor of shape N x C x D x H x W: batch size, channels, depth, height, width

4.3: Tabular data

- We can use np.loadtext() to read a CSV file, using the skiprows=1 argument to omit a header row
- Three kinds of numerical values:
    - **Continuous values:** strictly ordered and a difference between values has a strict meaning, eg. weight, distance, measurements, time
    - **Ordinal values:** strict ordering but no fixed relationship between values, eg. ordering a small, medium, or large drink
    - **Categorical values:** have neither ordering nor numerical meaning to their values, eg. assigning water = 1, coffee = 2, juice = 3
- We typically extract the target column from the input data and keep it in a separate tensor so that the ground truth isn't used as input to the model
- Ground truth labels can be represented as continuous variables (ie. kept as real numbers) or **one-hot encodings** (ie. vectors with all elements set to 0 except the index of the score)

4.4: Working with time series

- A 1D, multichannel dataset can be transformed into a 2D, multichannel dataset by separating the date and hour of each sample into separate axes

4.5 Representing text

- Networks operate on text at two levels:
    - Character level: processing one character at a time
    - Word level: individual words are the finest-grained entities seen by the network
- Regardless of which level is used, one-hot encoding can be used to encode the information into tensor form
- For character level one-hot encoding, each character in the text is represented as a zeros vector of length 128 (assuming UTF-8) with a 1 at the index corresponding to the current character
- For word level one-hot encoding, each word in the text is represented as a zeros vector of length equal to the number of unique words in the text. We put a 1 at the index corresponding to the current word.
- Alternatively, to compress the encoding, we could create an **embedding**: an effective way to map individual words to floating-point numbers.