Deep Learning

Mairi Hallman

May 2024

1 Introduction

2 A Brief Overview of Tensors

You are likely familiar with scalars, vectors, and matrices. These can be thought of as analagous data structures in zero, one, and two-dimensions, respectively. When generalizing to N dimensions, we refer to these collectively as tensors. A scalar is a zero-order tensor, a vector is a first-order tensor, and a matrix is a second-order tensor. A third-order tensor can be visualized as a stack of matrices. A fourth-order tensor would then be a vector of third order tensors. A fifth-order tensor is a matrix of third-order tensors... and so on.

2.1 Tensor Products

Tensor additon and subtraction are self-explanatory if matrix addition and subtraction are understood. The same cannot be said for tensor products. Below is an overview of tensor products necessary for the decompositions that will be presented in the next section.

2.1.1 Outer Product $\circ$

A tensor $T^{(N)}$ can be expressed as a product of N vectors. This is called the outer product (denoted $\circ$).

$$T^{(N)} = u_1 \circ u_2 \circ \cdots \circ u_N \tag{1}$$

2.1.2 Kronecker Product $\otimes$

The Konecker product of two matrices A and B, their Kronecker product is a matrix of the products of each element in A and the entire matrix B.

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & am2B & \dots & a_{mn}B \end{bmatrix} \tag{2}$$

2.1.3 Khatri-Rao Product $\odot$

The Khatri-Rao product of two matrices A and B, each with the same number of columns, is a matrix composed of the Kronecker products of the columns in matrix A and the columns in matrix B with the same indices.

$$A^{:\times n} \odot B^{:\times n} = \begin{bmatrix} a_{:,1} \otimes b_{:,1} & a_{:,2} \otimes b_{:,2} & \dots & a_{:,n} \otimes b_{:,n} \end{bmatrix} \tag{3}$$

2.1.4 Hadamard Product $*$

The Hadamard product of two matrices A and B of the same dimensions is a matrix formed of the products of the elements in A and B with the same indices.

$$A^{m\times n} * B^{m\times n} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \dots & a_{1n}b_{1n} \\ a_{21}b_{21} & a_{22}b_{22} & \dots & a_{2n}b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{m1} & a_{m2}b_{m2} & \dots & a_{mn}b_{mn} \end{bmatrix} \tag{4}$$

2.2 Tensor Decompositions

2.2.1 CP Decomposition

The canonical polyadic, or CP Decomposition, decomposes a tensor into vectors.

$$T = \sum_{r-1}^{R} u_r^{(1)} \circ u_r^{(2)} \circ \dots u_r^{(R)} \tag{5}$$

2.2.2 Tucker Decomposition

The Tucker decomposition decomposes a tensor into a core tensor and factored matrices.

$$T = C \prod_{n=1}^{N} A^n \tag{6}$$

where C is the core tensor.

2.2.3 Tensor Train

The tensor train decomposition decomposes a tensor into a product of third-order tensors. It is used when a tensor is too large for the CP decomposition to be practical.

3 Selecting A Network Architecture

4 Convolutional Neural Networks

4.1 What Is Convolution?

Many young people who are active on social media use filters on their pictures before posting them. These filters apply different effects to the photo, such as blurring, making the image black and white, or adding cartoon-like effects. When you apply a filter to a photo, you are using convolution. Convolution is an operation that takes the aggregation of the element-wise product of a tensor of input data and a (typically smaller) tensor, called a kernel. This produces a feature map. In the context of image processing, the original image is the input data, the filter is the kernel, and the filtered image is the feature map.

This is an example of discrete convolution, which is the case that we will be primarily concerned with.

In one dimension, discrete convolution is denoted

$$\underbrace{s(i)}_{\substack{\text{feature}\\ \text{map}}} = (x * y)(i) = \sum_m \underbrace{x(m)}_{\substack{\text{input}\\ \text{data}}} \underbrace{y(i-m)}_{\text{kernel}} \tag{7}$$

One of the most common applications of CNNs is in digital image processing. In the case of a black and white image, the input data is a two-dimensional tensor. Discrete convolution in two dimensions can be performed as follows [1].

$$s(i,j) = (x * y)(i,j) \sum_m \sum_n x(i,j) y(i-m, j-n) \tag{8}$$

Since convolution is commutative, equation 9 also holds.

$$s(i,j) = (y * x)(i,j) \sum_m \sum_n x(i-m, j-n) y(i,j) \tag{9}$$

Two-dimensional discrete convolution is equivalent to reversing the row and column indices of the kernel, performing element-wise multiplication, and taking the sum of the products. In the context of deep learning, the term "convolution" often refers to a similar operation called cross-correlation (equation). This is equivalent to convolution without the reversing of the matrix indices [1].

$$s(i,j) = (y * x)(i,j) \sum_m \sum_n x(i+m, j+n) y(i,j) \tag{10}$$

4.2 The Convolutional Layer

The convolutional layers used in a CNN have three steps [1].

1. Several parallel convolutions yield a set of linear activations.

2. Each linear activation function is run through a non-linear activation function, such as ReLU. This introduces non-linearity and gives the network more flexibility.

3. A pooling function replaces the output at a given location with a summary statistic of nearby outputs. One common pooling function is the max pooling function, which provides the maximum output within a rectangular neighbourhood. Pooling is useful when we only care about whether a feature is present and not its specific location. It is also helpful when processing different-sized inputs.

4.3 Image Classification Example

5 Recurrent Neural Networks

6 Generative Models

6.1 Generative Adversarial Neural Networks

6.2 Variational Autoencoders

6.3 Quantization