Course Notes: Deep Learning for Visual Computing

Peter Wonka

August 28, 2021

Contents

1 Tensors as Data Structure

1.1 Definition

- A **tensor** is a multi-dimensional array (of numbers)
  - This is a computer science definition focusing on the data structure
  - In mathematics, physics, mechanics, ... there may be an additional semantic meaning
  - Core data structure for machine learning
  - Google's **Tensorflow** has tensor in the name
- The **rank** of a tensor denotes the number of dimensions
  - This is different from matrix rank
  - This is different from the number of entries which is also called the dimension
  - A dimension is often called an **axis**
  - Python: `.ndim`

1.2 Scalars

- Scalars are 0D tensors
- E.g., $a, b, x \in \mathbb{R}$

1.3 Vectors

- Vectors are 1D tensors
- E.g., $\mathbf{x} \in \mathbb{R}^4$
  - 4-dimensional vector
  - tensor of rank (dimension) 1
  - tensor with 1 axis

```
1 import numpy as np
2 x = np.array([7, 3, 5, 11])
3 print(x)
4 print(x.ndim)
```

1.4 Matrices

- Matrices are 2D tensors
- Matrices have 2 axis (rows and columns)
- E.g., $\mathbf{A} \in \mathbb{R}^{2\times 3}$

```
1 import numpy as np
2 A = np.array([ [1, 2, 3], [4, 5, 6]])
3 print(A)
4 print(A.ndim)
```

1.5 Higher-dimensional tensors

- Images are 3D tensors
- RGB image $\mathbf{I} \in \mathbb{R}^{w \times h \times 3}$
- In CNNs we typically have images with many **channels** c
  - $\mathbf{I} \in \mathbb{R}^{w \times h \times c}$, where c can be up to a few thousand
- In deep learning we generally manipulate 0D to 5D tensors
- A **batch** (=sequence) of images with batch size n is a 4D tensor
  - $\{\mathbf{I}_j\} \in \mathbb{R}^{w \times h \times c \times n}$ or $\in \mathbb{R}^{n \times w \times h \times c}$ or $\in \mathbb{R}^{n \times c \times w \times h}$ or ...
- Simpler notation: $I(W, H, C)$
- PyTorch example: input size (N, C_{in}, H, W), output size $(N, C_{out}, H_{out}, W_{out})$
- Video: $T(samples, frames, height, width, channels)$

1.6 Python Commands

```
1 import numpy as np
2 A = np.random.rand(3,2,4)
3
4 print(A)
5 print(A.ndim) # the rank of the tensor = 3
6 print(A.shape) # vector describing the dimensions along each axis
7 print(A.dtype) # data type of the tensor
```

1.7 Manipulating Tensors

```
import numpy as np
A = np.random.rand(300,200,100)
B = A[20:50, :, :] # Slice notation
C = A[10:, :, 20:-20]

print( B.shape )
print( C.shape )

A = np.array([ [9, 9, 9], [-4, 5, 0]])
print( A )
x = np.array([7, 1, 0.1])
print( np.dot(A, x) ) #matrix vector multiplication

#checkout dot, matmul, +, -, /, *

print( np.maximum(A, 2) ) # elementwise max
```

1.8 Broadcasting

- For example: elementwise addition + of two tensors
- *Broadcasting* replicates data of a smaller tensor to make it compatible with the operation
  - Axes (broadcast axis) are added to the smaller tenstor to match the larger tensor
  - The smaller tensor is repeated alongside the new axes
- Broadcasting works for two tensors of size:
  - $(a, b, c, ..., n, n+1, ..., m)$ and $(n, n+1, ..., m)$
  - Broadcasting will happen for axes $a, ..., n-1$

1.9 Broadcasting Example

```
1 import numpy as np
2 A = np.random.rand(5,4)
3 x = np.array([1, 1, 1, 100])
4 print(A.shape)
5 print(x.shape)
6 print( A + x)
```

1.10 Tensor Reshaping

```
1 import numpy as np
2 A = np.array([[1,2,3], [4,5,6]])
3 print(A)
4 print(A.reshape(6,1))
5 print(A.reshape(3,2))
6 print(np.transpose(A))
```