भारतीय प्रौद्योगिकी संस्थान तिरुपति
IIT TIRUPATI

Basics of Tensors and Simple Example

Dr. Konda Reddy Mopuri
kmopuri@iittp.ac.in
Dept. of CSE, IIT Tirupati
Aug-Dec 2021

Tensor

1. Generalized matrix

Tensor

1. Generalized matrix
2. Finite table of data

Tensor

1. Generalized matrix
2. Finite table of data
3. Indexed along discrete dimensions

Tensor

1. 0D tensor - scalar

Tensor

1. 0D tensor - scalar
2. 1D tensor - vector (e.g. sound sample)

Tensor

1. 0D tensor - scalar
2. 1D tensor - vector (e.g. sound sample)
3. 2D tensor - matrix (e.g. gray-scale image)

Tensor

1. 0D tensor - scalar
2. 1D tensor - vector (e.g. sound sample)
3. 2D tensor - matrix (e.g. gray-scale image)
4. 3D tensor $\rightarrow$ vector of identically sized matrices (e.g. RGB image)

Tensor

1. 0D tensor - scalar
2. 1D tensor - vector (e.g. sound sample)
3. 2D tensor - matrix (e.g. gray-scale image)
4. 3D tensor $\rightarrow$ vector of identically sized matrices (e.g. RGB image)
5. 4D tensor $\rightarrow$ matrix of identically sized matrices or a sequence of 3D tensors (e.g. sequence of RGB images)

Tensor

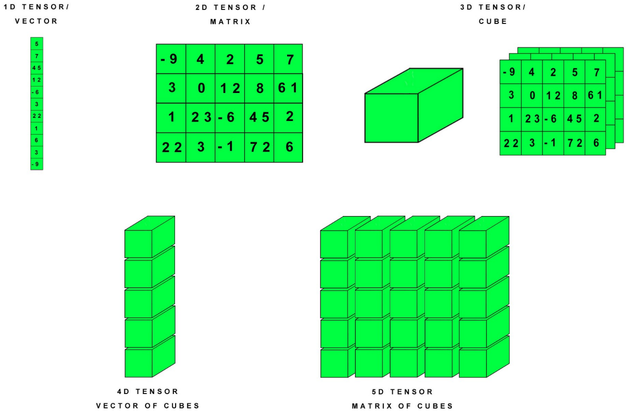

Figure Credits: datacamp.com

Tensors are used to

1. - Encoding signals (e.g. text, speech, image, etc.)
   - Internal states and parameters of the model

Tensors are used to

1. - Encoding signals (e.g. text, speech, image, etc.)
   - Internal states and parameters of the model
2. Operating on data through this constrained structure allows CPUs and GPUs to operate at their near peak performance.

PyTorch's main features

1. Efficient tensor manipulations on CPU/GPU
   - Standard LA and DL specific operations

PyTorch's main features

1. Efficient tensor manipulations on CPU/GPU
   - Standard LA and DL specific operations
2. Autograd: automatic on-the-fly differentiation

PyTorch's main features

1. Efficient tensor manipulations on CPU/GPU
   - Standard LA and DL specific operations
2. Autograd: automatic on-the-fly differentiation
3. Optimizers
   - Variety of optimizers (e.g. SGD, Adam, etc.); Hassle-free to apply

PyTorch's main features

1. Efficient tensor manipulations on CPU/GPU
   - Standard LA and DL specific operations
2. Autograd: automatic on-the-fly differentiation
3. Optimizers
   - Variety of optimizers (e.g. SGD, Adam, etc.); Hassle-free to apply
4. Data i/o
   - Load a data sample or datasets, etc.

Tensor Basics

▸ Colab Notebook: Tensor basics

Example: Linear Regression

Linear Regression

1. Given a set of data points $(x_n, y_n) \in \mathcal{R} \times \mathcal{R}, n = 1, 2, \ldots N$

Linear Regression

1. Given a set of data points $(x_n, y_n) \in \mathcal{R} \times \mathcal{R}, n = 1, 2, \dots N$
2. Finding the best line that fits the data, $f(x; a, b) = ax + b$

Linear Regression

1. Given a set of data points $(x_n, y_n) \in \mathcal{R} \times \mathcal{R}, n = 1, 2, \ldots N$
2. Finding the best line that fits the data, $f(x; a, b) = ax + b$
3. i.e., minimizes the mean squared error (MSE), $\arg\min_{a,b} \frac{1}{N} \Sigma_{i=1}^{N} (ax_n + b - y_n)^2$

Linear Regression

1. $a = \frac{\Sigma_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\Sigma_{i=1}^{N}(x_i - \bar{x})^2}$

Linear Regression

1. $a = \frac{\Sigma_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\Sigma_{i=1}^{N}(x_i - \bar{x})^2}$
2. $b = \bar{y} - a\bar{x}$

Linear Regression

▸ Colab Notebook: Linear Regression

Tensors can be

- torch.float16, torch.float32, torch.float64
- torch.uint8
- torch.int8, torch.int16, torch.int32, torch.int64

Tensors can be

- Located either in CPU or in GPU
- Operations are performed only by that device in whose memory the tensor is stored