Pytorch

Ramex

June 2024

1 Tensors

A Tensor is like an n-dimensional array (multi-dimensional array) in numpy, but with additional features. A tensor can be created from a list or a numpy array. The tensor can be converted to a numpy array using the `numpy()` method. The tensor itself can be used for gpu computations.

1.1 Initializing a Tensor

A tensor can be initialized by a number of methods, for example:

- 1. Directly from data
  `data = [[1,2,3],[4,5,6]],`
  `data_ = torch.tensor(data)`
- 2. From a numpy array
  `data = np.array(data),`
  `data_x = torch.tensor(data)`
- 3. From another tensor (creates a copy of the tensor with ones only)
  `torch.ones_like(x_data)`
- 4. Random or constant values (this takes the shape as input)
  `x_ones = torch.ones_like(x_data)`

1.2 Attributes of a Tensor

A tensor has the following attributes:

- 1. `shape` : The shape of the tensor
- 2. `dtype` : The data type of the tensor
- 3. `device` : The device on which the tensor is stored
- 4. `size` : The number of elements in the tensor
- 5. `numel` : The number of elements in the tensor

- 6. `T` : The transposed tensor
- 7. `contiguous` : The contiguous tensor
- 8. `view` : The view of the tensor
- 9. `requires_grad` : The gradient required for the tensor
- 10. `grad` : The gradient of the tensor
- more ... ⇒ Pytorch Documentation

1.3 Indexing and Slicing

Indexing and slicing works the same as in numpy. For example:

- 1. `x[0]` : The first element of the tensor
- 2. `x[0,0]` : The first element of the first row
- 3. `x[0,:]` : The first row of the tensor
- 4. `x[:,0]` : The first column of the tensor
- 5. `x[0:2,0:2]` : The first two rows and columns of the tensor

1.4 Joining tensors

Tensors can be joined using the `torch.cat()` method. For example:

- 1. `torch.cat([x,y], dim=0)` : Concatenates the tensors along the rows
- 2. `torch.cat([x,y], dim=1)` : Concatenates the tensors along the columns
- 3. `torch.stack([x,y], dim=0)` : Stacks the tensors along the rows
- 4. `torch.stack([x,y], dim=1)` : Stacks the tensors along the columns

1.5 Single-element tensors

A single-element tensor is a tensor with one element. For example:

- 1. `x.item()` : Returns the value of the tensor as a python number
- 2. `x.tolist()` : Returns the value of the tensor as a python list
- 3. `x.numpy()` : Returns the value of the tensor as a numpy array
- 4. `x.to(device)` : Moves the tensor to the specified device

1.6 Tensor to NumPy array

A tensor can be converted to a numpy array using the `numpy()` method. Example:

- 1. `x.numpy()` : Converts the tensor to a numpy array
- 2. `x.cpu().numpy()` : Converts the tensor to a numpy array on the cpu
- 3. `x.cuda().numpy()` : Converts the tensor to a numpy array on the gpu

2 Datasets & DataLoaders

2.1 Loading a Dataset

A dataset can be loaded using the `torch.utils.data.Dataset` class.

2.2 Iterating and Visualizing the Dataset

A dataset can be iterated over using a for loop. For example:

3 Transforms

Data does not always come in its final processed form that is required for training machine learning algorithms. We use transforms to perform some manipulation of the data and make it suitable for training.

The FashionMNIST features are in PIL Image format, and the labels are integers. For training, we need the features as normalized tensors, and the labels as one-hot encoded tensors. To make these transformations, we use ToTensor and Lambda.