PyTorch Cheat Sheet

V2021.01.16

(Dr Yan Xu)

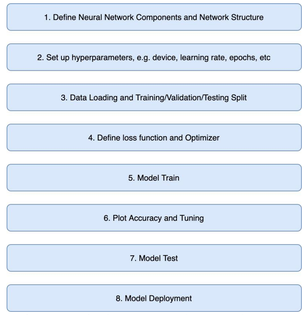
What is PyTorch?

- NumPy on GPU
- Automatic differentiation library
- State of the art neural network library
- Utility libraries for vision, text and audio
- Open source machine learning framework

Torch.Tensor

- Data Type:
  - FloatTensor
  - BoolTensor
  - LongTensor
- Tensor Properties:
  - data, dtype, device, requires_grad
- Data Type Transformation:
  - float(), bool(), int()
- Numpy Array & Python List:
  - torch.tensor()
  - numpy()
  - torch.from_numpy()
  - tolist()
- Construction:
  - torch.rand(size=(3, 3))
  - torch.randn(size=(3, 3))
  - torch.arange(start=1, end=10, step=2)
  - torch.linspace(3, 10, steps=3)
  - torch.logspace(-4, 0, steps=5, base=10)
- Data Selection:
  - Same with numpy, e.g. tensor[:, 0]
- Functions:
  - max, argmax, min, argmin
  - mean, median, mode, var, quantile, sum
  - dot, mm, inverse, cholesky, svd
- Transformation:
  - view, reshape
  - unsqueeze *[add dim]*, squeeze *[remove dim if 1]*
  - cat, transpose, permute

General Neural Model Steps



- Code example: basic neural network

Weight Initialization

- Parameters in nn.Module have default initialization method, e.g.
  - Linear Layer: pytorch github
  - Conv Layer: pytorch github
  - BatchNorm Layer: pytorch github
- Customize Initialization
  - nn.init.uniform_(tensor, a=0.0, b=1.0)
  - nn.init.kaiming_uniform_()
  - nn.init.constant_(tensor, val)

Reproducibility

- torch.manual_seed(seed)
- numpy.random.seed(seed)
- random.seed(seed)
- torch.backends.cudnn.benchmark = False
- torch.set_deterministic(True)

Debugging: TensorBoard

tensorboard --logdir=PATH/LOG

- Plot loss time series
- Plot Images/transformed images
- Compare hyper-parameters
- Code example: github

DataSet & DataLoader

- DataSet: A class representing a dataset
  - It's common to write own DataSet
  - We could directly access DataSet by [i], but it is much common to access by Data Loader
- DataSet Code Template

```
class MyDataSet(Dataset):
    def __init__(self, csv_file, transform=None):
        self.data = pd.read_csv(csv_file)
        self.feature, self.target = split(self.data)
        if transform:
            self.feature = transform(self.feature)

    def __len__(self):
        return len(self.target)

    def __getitem__(self, idx):
        return self.feature[idx], self.target[idx]
```

- Common Operations on DataSet:
  - random_split()
  - RandomSampler()
  - Subset()
- DataLoader: a smart iterator
  - batch_size
  - shuffle
  - collate_fn

```
for features, target in data_loader:
    pred = model(features)
    cost = criterion(target, pred)
```

- No need to define own Data Loader

Standard Loss Functions

- Regression
  - nn.MSELoss*(Input, Target)*
    - Input: (N) where value is any float number
    - Target: (N) where value is any float number
  - nn.L1Loss*(Input, Target)*
    - Input: (N) where value is any float number
    - Target: (N) where value is any float number
- Classification
  - ***Multi-class:*** *CrossEntropyLoss(Input, Target)*
    - Input: (N,C) where C = number of classes
    - Target: (N) where each value is [0, C−1]
  - ***Binary-class:*** BCEWithLogitsLoss*(Input, Target)*
    - Input: (N) where value is any float number
    - Target: (N) where each value is [0, 1]
  - *note: these two loss functions include sigmoid function already.*

Image Classification

- torch.nn
  - nn.Conv2d
    - E.g. Conv2d(in_channels=3, out_channels=8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  - nn.MaxPool2d
    - E.g. MaxPool2d(kernel_size=(2, 2), stride=(2, 2))
  - nn.BatchNorm2d
  - nn.Sequential
- torchvision.datasets
- torchvision.models
  - VGG
  - Inception v3
- Transfer learning: code
  - model = models.vgg16() # random parameters
  - model = models.vgg16(pretrained=True) # reuse initial parameters
  - model.parameters().requires_grad = False # fix parameters
- torchvision.transforms
  - transforms.RandomAffine
  - transforms.RandomRotation
  - transforms.ToTensor
  - transforms.Normalize
- Pytorch VGG: code

Object Detection (YOLO)

- Location Encoding (target encoding)
- Neural Network
  - DarkNet, RestNet, etc
- Accuracy Measurement
  - IOU (intersection_over_union)
  - mAP (mean_average_precision)
- Non Max Suppression (code)
- YOLO Cost (classification cost & box cost)
- Image/Box Transformation
- Not much support in Pytorch yet
- Pytorch 3rd Lib: github

Neural Style Transfer

- Motivation: fix Neural Network parameters and optimize on the input image
- Cost Function:
  - Image similarity & style similarity
- Code example: github

Basic Natural Language Processing

- torchtext.vocab.Vocab
  - .freqs: the frequencies of tokens
  - .stoi: mapping token strings to numerical identifiers
  - .itos: numerical identifiers to strings
- torchtext.data.Dataset
  - It is different from general dataset, because it allows for each column containing sequence of features.
  - It must have two functions: split() and iters()
- torchtext.data.TabularDataset
- torchtext.data.Field
  - The concept is kind of strange, but it primarily tells whether a column is a sequence or singular value.
- Overall, torchtext.data.dataset has a strange code flavor, an example blow:

```
# create Field objects
AUTHOR = data.Field()
AGE = data.Field()
TWEET = data.Field()
LOCATION = data.Field()

# create tuples representing the columns
fields = [
  ('author', AUTHOR),
  ('location', LOCATION),
  (None, None), # ignore age column
  ('tweet', TWEET)
]

# load the dataset in json format
train_ds, valid_ds, test_ds = data.TabularDataset.splits(
   path = 'data',
   train = 'train.tsv',
   validation = 'valid.tsv',
   test = 'test.tsv',
   format = 'tsv',
   fields = fields,
   skip_header = True
)
```

- Tokenizer
  - text.split()
  - torchtext.data.get_tokenizer(text)
  - spacy tokenizer
- common NLP Datasets
  - E.g. torchtext.datasets.IMDB
- torch.nn
  - nn.RNN(input_size, hidden_size, num_layers)
  - nn.GRU(input_size, hidden_size, num_layers)
  - nn.LSTM(input_size, hidden_size, num_layers)
- Example NLP POC
  - bag-of-word model
  - Embedding (glove) model
  - GRU model

Basic Audio Processing

- torchaudio.load()
  - datwaveform, sample_rate = torchaudio.load('example.mp3')
- torchaudio.transforms
  - torchaudio.transforms.MelSpectrogram (*most popular*)
    - *sample_rate: int*
    - *win_length: int*
    - *hop_length: int*
    - *n_fft: int*
    - *F_min: float*
    - *F_max: float*
  - torchaudio.transforms.Spectrogram
  - torchaudio.transforms.MelScale
- torchtext.datasets
- torchaudio.models
- Example Audio ML POC code
  - Audio Feature Extraction
  - Audio + RestNet