PyTorch Tutorial

Yunfei Teng

yt1208@nyu.edu

Department of Electrical and Computer Engineering
New York University Tandon School of Engineering

March 1, 2018

Outline

In this tutorial, everything is arranged in the recommended order of reading. Some are supplementary materials to enlarge your knowledge of deep learning. If you'd like to selectively pick the most important things, read red parts.

1. Artificial Neural Networks(ANNs)
   - What is a neural network?
   - How to build a neural network?
   - Why is it hard to train a neural network?
2. PyTorch
   - Introduction to PyTorch and its advantage
   - Autograd mechanism
3. Let's do it!
4. Extended reading

What is a neural network?

- "...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.
  In "Neural Network Primer: Part I" by Maureen Caudill, AI Expert, Feb. 1989.
- It's similar to calculus of variations. Extended Reading: `https://en.wikipedia.org/wiki/Calculus_of_variations`.
- It can be considered as a tool which is able to approximate any complex function("universal approximators").
- All the nodes of neural networks should be differentiable due to Backpropagation Algorithm.

How to build a neural network?

What if you'd like to start from scratch?

- Think of the concept of Object-Oriented Programming(OOP). How to decompose a large neural network into small parts?
- One possible way to build it: Neuron $\rightarrow$ Layer $\rightarrow$ Network. If you are interested, you can check this online blog for more details: C++ Implementation of Neural Netowrks: `https://takinginitiative.wordpress.com/2008/04/23/basic-neural-network-tutorial-c-implementation-and-source`

In PyTorch the smallest unit is layer and neurons are implicitly included in layers. Here we will discuss three important classes in PyTorch: torch.utils.data, torch.nn and torch.optim.

torch.utils.data: Load and process data.
Related document: `http://pytorch.org/docs/master/data.html`

torch.optim: Using different optimization methods decides how neural networks make use of these gradients.
Related document: `http://pytorch.org/docs/master/optim.html`

How to build a neural network?

torch.nn class:

- Basic layers:
  - Linear layers: Apply a linear transformation over an input.
  - Convolution layers: Apply a convolution over an input.
  - Activation layers: Introduce non-linearity.
- Other important layers:
  - Pooling layers: Reduce the spatial size of the representation.
  - Normalization layers: Reduce the Covariate Shift.
  - Dropout layers: Prevent overfitting.
  - Data parallel layers: Run data in parallel.
- Loss functions: Define cost for difference between output and target.

Related document: `http://pytorch.org/docs/master/nn.html`

In PyTorch, each layer has its own parameters. These parameters belong to autograd.Variable class with attributes of data and grad. A layer's weights and gradients are saved in data and grad respectively.

Why is it hard to train a neural network?

- It's non-convex? Yes.
  New optimization methods and architectures are invented to alleviate this problem.
  - Entropy-SGD: `https://arxiv.org/pdf/1611.01838.pdf`
  - ResNet: `https://arxiv.org/pdf/1512.03385.pdf`
- It's a black box? Maybe.
  New visualization tools helped us to understand it.
  - Visualizing and Understanding Convolutional Networks: `https://cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf`
  - Visual BackProp: `https://arxiv.org/pdf/1611.05418.pdf`
- The reasons are complex: Learning rate, optimizer, initialization, data, pipeline, model, architecture, etc. All these factors count for results.

Introduction to PyTorch and its advantage

- It's a **Python** first package.
- It's not only designed for neural networks, but also works as a **GPU-ready Tensor library**.
- Compared with Tensorflow and other packages, PyTorch is more friendly to researchers since the codes are more **concise**.
- The graph is **built on the fly**, so you can change the architecture at any time!
- You can **switch between CPU and GPU** easily.
- What's more? The best way to learn Python is: 1)Read Documents 2) Practice by solving problems 3) Follow an **open-source project**. Learning PyTorch will guarantee you all of these things.

Autograd mechanism

- Remember Neural Networks should be differentiable? It would be wonderful if we have something doing this job automatically. PyTorch can do it!
- PyTorch autograd tutorial: `http://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html`
- Autograd project from HIPS: `https://github.com/HIPS/autograd`
- More about Tensor and autograd functions `https://github.com/jcjohnson/pytorch-examples`
- Differentiable Memory: `https://deepmind.com/blog/differentiable-neural-computers/`

Let's do it!

You can download the codes of this tutorial from my Github:
`https://github.com/yunfei-teng/PyTorch-Tutorial`.

1. Calculating cosine similarity of vectors
   -Write better algorithm.
2. Regression for polynomial functions
   -Understand why neural networks are "universal approximators".
3. MNIST digit classification
   -Learn to design a complete project.

You can find support from PyTorch websites:

- For more examples, go to official Github:
  `https://github.com/pytorch/examples`
- For more specific explanations, go to official documents:
  `http://pytorch.org/docs/master/index.html`

Extended reading

- Online Book of Deep Learning
  `http://neuralnetworksanddeeplearning.com`
- Back-Propagation
  `https://en.wikipedia.org/wiki/Backpropagation`
- Convolutional Neural Networks
  `http://cs231n.github.io/convolutional-networks/`
- PyTorch in 60 minutes:
  `http://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html`