Introduction to PyTorch

Joshua Yao-Yu Lin (林曜宇)
University of Illinois at Urbana-Champaign

[@HAL training 2021.9.29]
Joshua Yao-Yu Lin

- My research spans a wide range of Machine Learning application in Astrophysics
- Research Interest: dark matter, supermassive black holes, neuroscience, **machine learning**
- Before Joining UIUC, I got my MS at NTU, and BS at NTHU (All in physics).
- ML intern experience: Simons Foundation, Google Research
- I’ve used PyTorch for most of my deep learning projects!

Physics PhD student (2016-Present)
University of Illinois at Urbana-Champaign
Agenda

● Overview of PyTorch & Deep Learning
● Pytorch Basics
● Train a Convolutional neural networks to classify MNIST data
● Train a Variational Autoencoder to generate new MNIST data
● Q & A section

https://github.com/joshualin24/NCSA_pytorch_tutorial
Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.
CLASSICAL MACHINE LEARNING

**SUPERVISED**
- Data is pre-categorized or numerical
  - Predict a category
    - **CLASSIFICATION**
      - "Divide the socks by color"
  - Predict a number
    - **REGRESSION**
      - "Divide the ties by length"

**UNSUPERVISED**
- Data is not labeled in any way
  - Divide by similarity
    - **CLUSTERING**
      - "Split up similar clothing into stacks"
  - Identify sequences
    - **ASSOCIATION**
      - "Find what clothes I often wear together"
  - Find hidden dependencies
    - **DIMENSION REDUCTION (generalization)**
      - "Make the best outfits from the given clothes"

Credits: https://vas3k.com/blog/machine_learning/
What is PyTorch?

- **Open source** machine learning library
- Developed by Facebook's AI Research lab
- It leverages the power of GPUs
- Automatic computation of gradients
- Makes it easier to test and develop new ideas.
Why PyTorch?
Why PyTorch?

- It is pythonic - concise, close to Python conventions
- Strong GPU support
- Autograd - automatic differentiation
- Many algorithms and components are already implemented
- Similar to NumPy
Why PyTorch?

**Computation Graph**

```
x * y  \rightarrow  a
+     \rightarrow  b
\sum  \rightarrow  c
```

**Numpy**

```python
import numpy as np
np.random.seed(0)
N, D = 3, 4
x = np.random.randn(N, D)
y = np.random.randn(N, D)
z = np.random.randn(N, D)
a = x * y
b = a + z
c = np.sum(b)
```

**Tensorflow**

```python
import numpy as np
import tensorflow as tf
N, D = 3, 4
x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)
z = tf.placeholder(tf.float32)
a = x * y
b = a + z
c = tf.reduce_sum(b)
```

**PyTorch**

```python
import torch
N, D = 3, 4
x = torch.randn((N, D), requires_grad=True)
y = torch.randn((N, D), requires_grad=True)
z = torch.randn((N, D), requires_grad=True)
a = x * y
b = a + z
c = torch.sum(b)
c.backward()
```
Pytorch Tensor

- Basic block of pytorch
- Similar to numpy array
- Easy to operate on GPU/CPU
- Good for building neural networks

```python
import torch
x = torch.tensor([[1,2,3],[4,5,6]])
y = torch.tensor([[7,8,9],[10,11,12]])
f = 2*x + y
print(f)
tensor([[ 9, 12, 15],
        [18, 21, 24]])
```
Demo: pytorch basics

MNIST for image classification
MNIST in pixels
Introduction to neural networks with pytorch
Convolution & Feature Map generation

- Intermediate output inside the hidden layers of neural networks
- “Down sample”
- Single input can induce multiple feature maps (with different kernels)
Tanh
\[ \tanh(x) \]

ReLU
\[ \max(0, x) \]

Sigmoid
\[ \sigma(x) = \frac{1}{1+e^{-x}} \]

Linear
\[ f(x) = x \]
Max pooling

<table>
<thead>
<tr>
<th></th>
<th>3</th>
<th>1</th>
<th>7</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>4</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Max Pooled

Kernel/Filter - 2x2
Stride 2

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>
Loss function: Cross-Entropy

\[ H(p, q) = - \sum_{x \in \text{classes}} p(x) \log q(x) \]
Loss function: Cross-Entropy

\[ D(S, L) = - \sum_i L_i \log(s_i) \]
Backpropagation
Introduction to neural networks with pytorch

Image

Convolution
padding = 1,
kernel = 3x3,
stride = 1
+ ReLU

32 x 28 x 28

32 x 14 x 14

Convolution
padding = 1,
kernel = 3x3,
stride = 1
+ ReLU

64 x 14 x 14

Max pooling
Kernel = 2x2,
Stride = 2

64 x 7 x 7

Max pooling
Kernel = 2x2,
Stride = 2

3136 x 128

Flatten

128 x 10
Let’s build a neural network together!

https://bit.ly/3EZ7mMR
Generative models

- Neural networks are not only for classification (supervised learning)!
- Generative models: creating new data with deep neural networks
- Example of generative models:
  - Generative adversarial networks (GAN)
  - Variational Autoencoder (VAE)
  - Normalizing Flow
Variational Autoencoder (VAE)

Input $\xRightarrow{\text{Ideally they are identical.}} x \approx x'$

Encoder $g_\phi$

Decoder $f_\theta$

Bottleneck!

An compressed low dimensional representation of the input.
Q&A

Feel free to reach out to me:

joshualin24@gmail.com

Twitter: joshualin24

Github: joshualin24