


1. 概述

 PyTorch 是一个开源的机器学习框架，用于数据的表示和处理。Tensor (张量) 是一种多维矩阵。PyTorch 中所有神经网络的核心元素是 Autograd 软件包，该软件包可为所有有关 Tensor 的操作提供自动区分。



导入基础工具库

```
import torch
```

```
import torch.nn as nn
```

pytorch 中用于视觉的数据集、模型与变换

```
from torchvision import datasets, models, transforms
```

```
import torch.nn.functional as F # Function Collection
```

```
torch.randn(*size) # 随机张量
```

```
torch.Tensor(L) # 由 L 初始化的张量
```

```
tnsr.view(a, b, ...) # Tensor in size (a, b,...) transform
```

```
requires_grad=True # 计算梯度
```

2. 层 layer



nn.Linear(m, n)

从 m 到 n 神经元的全连接层 (或稠密层)



nn.Flatten()

将 tensor 的维度减少到 1 (展平矩阵成向量)



nn.Dropout(p=0.5)

在训练期间，将输入随机设置为 0 (随机失活)；有助于避免过拟合



nn.Embedding(m, n)

显示 n 维向量上大小为 m 的目录中的索引



nn.ConvXd(m, n, s)

X ∈ {1,2,3}

输入通道数 m，输出通道数 n，卷积核大小为 s 的 X 维卷积层



nn.MaxPoolXd(s)

池化核大小为 s 的 X 维池化层；X ∈ {1,2,3}



nn.BatchNormXd(n)

将具有 n 个特征的 X 维输入批次标准化；X ∈ {1,2,3}



nn.RNN/LSTM/GRU

递归网络将一层的神经元连接到同一层或前一层的神经元

3. 加载数据

一条样本可以用从 Dataset (Features, Label) 元组构建的序列) 继承而来的类来初始化表示。使用 DataLoader，可以分批完成加载。数据集通常分为训练数据 (通常 80%) 和测试数据 (通常 20%)。

```
from torch.utils.data import Dataset, TensorDataset, DataLoader, random_splits
train_data, test_data = random_split(TensorDataset(inps, tgts), [train_size, test_size])
train_loader = DataLoader(dataset=train_data, batch_size=16, shuffle=True)
```

4. 激活 activation

最常见的激活函数包括 ReLU、Sigmoid 和 Tanh。当然，也还有许多其他激活函数。

nn.ReLU() 是一个新的模块和序列模型，是 ReLU 函数的新版本。



nn.ReLU() oder **F.relu()**

输出介于 0 和 ∞ 之间，最常见的激活函数



nn.Sigmoid() oder **F.sigmoid()**

输出介于 0 和 1 之间，通常用于概率



nn.Tanh() oder **F.tanh()**

介于 -1 和 1 之间的问题，通常用于两个类

5. 定义模型

nn.Sequential

```
model = nn.Sequential(
```

```
    nn.Conv2D( , , )
```

```
    nn.ReLU()
```

```
    nn.MaxPool2D( )
```

```
    nn.Flatten()
```

```
    nn.Linear( , ))
```

class

```
class Net(nn.Module):
```

```
    def init ():
```

```
        super(Net, self).init ()
```

```
        self.conv = nn.Conv2D( , , )
```

```
        self.pool = nn.MaxPool2D( )
```

```
        self.fc = nn.Linear( , )
```

```
    def forward(self, x):
```

```
        x = self.pool(F.relu(self.conv(x)))
```

```
        x = x.view(-1, )
```

```
        x = self.fc(x)
```

```
        return x
```

```
model = Net()
```

在 Pytorch 中定义神经网络有几种方法

例如：

使用 nn.Sequential

使用 Class

使用两者的组合



扫码回复“工具库”

下载 最新 全套资料

6. 保存 / 加载模型

`model.state_dict()`: 通常只保存模型参数, 而不保存整个模型

```
model = torch.load('PATH') # Load Model
torch.save(model, 'PATH') # Save Model

torch.save(model.state_dict(), 'params.ckpt')
model.load_state_dict(torch.load('params.ckpt'))
```

7. GPU 训练

`model.to(device)`: 计算任务分别发送到各设备

```
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu') # 有 CUDA 支持的 GPU 可用
inputs, labels = data[0].to(device), data[1].to(device) # 已传送到 ID 为 0 的 GPU
```

8. 训练

各种计算误差的方法

```
nn.L1Loss # 平均绝对误差 (Mean Absolute Error)
nn.MSELoss # 均方差 (Mean Squared Error / L2Loss)
nn.CrossEntropyLoss # 交叉熵 (Cross-Entropy)
nn.BCELoss # 二元交叉熵 (Binary Cross-Entropy)
```

优化算法

`torch.optim` 优化算法是一种有效的梯度增强算法, 它可以有效地提高动态性能。

```
# Stochastic Gradient Descent
optim.SGD
```

```
# Adaptive Moment Estimation
optim.Adam
```

```
# Adaptive Gradient
optim.Adagrad
```

```
# Root Mean Square Prop
optim.RMSProp
```

```
import torch.optim as optim
```

```
loss_fn = nn.CrossEntropyLoss() # 损失函数
```

```
# 随机梯度下降用于优化
```

```
optimizer = optim.SGD(model.parameters(), lr = 0.001, momentum = 0.9)
```

```
for epoch in range(2): # 在训练集上遍历多次
    model.train() # 启用训练模式
    for i, data in enumerate(train_loader, 0):
```

```
        inputs, labels = data # 数据以 batch 个 [输入、标签] 的形式组织
        optimizer.zero_grad() # 梯度初始化为 0
        outputs = model(inputs) # 计算模型输出
        loss = loss_fn(outputs, labels)
        loss.backward() # 计算损失并传回
        optimizer.step() # 更新权重 / 学习率
```

9. 评估

根据数据指标评估模型的训练结果。评估目标不同, 使用的数据指标也随之变化。常用的指标包括: 准确度 (accuracy)、精确度 (precision)、召回率 (recall)、F1 或 BLEU。

```
model.eval() # 启用评估模式, 此处某些层的行为不同
```

```
torch.no_grad() # 禁用自动差分 / 求导, 减少内存需求
correct = 0 # 正确分类的样本数
total = 0 # 总分类样本数
```

```
model.eval()
with torch.no_grad():
    for data in test_loader:
        inputs, labels = data
        outputs = model(inputs)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0) # Batch
        correct += (predicted == labels).sum().item()

print('Accuracy: %s' % (correct/total))
```

PYTORCH



扫码回复“工具库”

下载最新全套资料

PyTorch 深度学习应用速查表

获取最新版 | <http://www.showmeai.tech/>

作者 | 韩信子 @ShowMeAI

设计 | 南乔 @ShowMeAI

参考 | Stefan Seegerer, Matthias Zurl

数据科学工具库速查表



NumPy 是 Python 数据科学计算的核心库，提供了高性能多维数组对象及处理数组的工具。使用以下语句导入 NumPy 库：

```
import numpy as np
```



SciPy 是基于 NumPy 创建的 Python 科学计算核心库，提供了众多数学算法与函数。



Pandas 是基于 NumPy 创建的 Python 库，为 Python 提供了易于使用的的数据结构和数据分析工具。使用以下语句导入：

```
import pandas as pd
```



Matplotlib 是 Python 的二维绘图库，用于生成符合出版质量或跨平台交互环境的各类图形。

```
import matplotlib.pyplot as plt
```



Seaborn 是基于 matplotlib 开发的高阶 Python 数据可视图库，用于绘制优雅、美观的统计图形。使用下列别名导入该库：

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```



Bokeh 是 Python 的交互式可视图库，用于生成在浏览器里显示的大规模数据集高性能可视图。Bokeh 的中间层通用 **bokeh.plotting** 界面主要为两个组件：数据与图示例。

```
from bokeh.plotting import figure
```

```
from bokeh.io import output_file, show
```



PySpark 是 Spark 的 Python API，允许 Python 调用 Spark 编程模型。Spark SQL 是 Apache Spark 处理结构化数据模块。

AI 垂直领域工具库速查表



Scikit-learn 是开源的 Python 库，通过统一的界面实现机器学习、预处理、交叉验证及可视化算法。



Keras 是强大、易用的深度学习库，基于 Theano 和 TensorFlow 提供了高阶神经网络 API，用于开发和评估深度学习模型。



“TensorFlow™ is an open source software library for numerical computation using data flow graphs.” **TensorFlow** 是 Google 公司开发的机器学习架构，兼顾灵活性和扩展性，既适合用于工业生产也适合用于科学研究。



PyTorch 是 Facebook 团队 2017 年初发布的深度学习框架，有利于研究人员、爱好者、小规模项目等快速搞出原型。**PyTorch** 也是 Python 程序员最容易上手的深度学习框架。



Hugging Face 以开源的 NLP 预训练模型库 **Transformers** 而广为人知，目前 GitHub Star 已超过 54000+。**Transformers** 提供 100+ 种语言的 32 种预训练语言模型，简单，强大，高性能，是新手入门的不二选择。



OpenCV 是一个跨平台计算机视觉库，由 C 函数 /C++ 类构成，提供了 Python、MATLAB 等语言的接口。**OpenCV** 实现了图像处理和计算机视觉领域的很多通用算法。

编程语言速查表



SQL 是管理关系数据库的结构化查询语言，包括数据的增删查改等。作为数据分析的必备技能、岗位 JD 的重要关键词，SQL 是技术及相关岗位同学一定要掌握的语言。



Python 编程语言简洁快速、入门简单且功能强大，拥有丰富的第三方库，已经成为大数据和人工智能领域的主流编程语言。

More...

AI 知识技能速查表



Jupyter Notebook 交互式计算环境，支持运行 40+ 种编程语言，可以用来编写漂亮的交互式文档。这个教程把常用的基础功能讲解得很清楚，对新手非常友好。



正则表达式 非常强大，能匹配很多规则的文本，常用于文本提取和爬虫处理。这也是一门令人难以捉摸的语言，字母、数字和符号堆在一起，像极了“火星文”。

More...



ShowMeAI 速查表 (©2021)

获取最新版 | <http://www.showmeai.tech/>

作者 | 韩信子 @ShowMeAI

设计 | 南乔 @ShowMeAI

数据科学工具库速查表

扫码回复“数据科学”

获取最新全套速查表

AI 垂直领域工具库速查表

扫码回复“工具库”

获取最新全套速查表

编程语言速查表

扫码回复“编程语言”

获取最新全套速查表

AI 知识技能速查表

扫码回复“知识技能”

获取最新全套速查表