



本书使用环境：Python 3.6+ 、Pytorch1.0+ 、TensorFlow1.5+，GPU 或 CPU
系统为：Linux 或 windows

本书代码及数据的网址：www.feiguyunai.com

如果您发现问题或有建议，可通过以下方式给我们反馈：

邮箱：wumg3000@163.com

微信：wumg3000

目录

前言

第一部分 基础篇

第 1 章 Numpy 基础

1.1 生成 Numpy 数组

1.1.1 从已有数据中创建数组

1.1.2 利用 random 模块生成数组

1.1.3 创建特定形状的多维数组

1.1.4 利用 arange、linspace 函数生成数组

1.2 获取元素

1.3 Numpy 的算术运算

1.3.1 对应元素相乘

1.3.2 点积运算

1.4 数组变形

1.4.1 更改数组的形状

1.4.2 合并数组

1.5 批量处理

1.6 通用函数

1.7 广播机制

1.8 小结

第 2 章 Pytorch 基础

2.1 为何选择 Pytorch?

2.2 安装配置

2.2.1 安装 CPU 版 Pytorch

2.2.2 安装 GPU 版 Pytorch

2.3 Jupyter Notebook 环境配置

2.4 Numpy 与 Tensor

2.4.1 Tensor 概述

2.4.2 创建 Tensor

2.4.3 修改 Tensor 形状

2.4.4 索引操作

2.4.5 广播机制

2.4.6 逐元素操作

2.4.7 归并操作

2.4.8 比较操作

2.4.9 矩阵操作

2.4.10 Pytorch 与 Numpy 比较

2.5 Tensor 与 Autograd

2.5.1 自动求导要点

2.5.2 计算图

2.5.3 标量反向传播

2.5.4 非标量反向传播

2.6 使用 Numpy 实现机器学习

2.7 使用 Tensor 及 autograd 实现机器学习

2.8 使用 TensorFlow 架构

2.9 小结

第 3 章 Pytorch 神经网络工具箱

3.1 神经网络核心组件

3.2 实现神经网络实例

3.2.1 背景说明

3.2.2 准备数据

3.2.3 可视化源数据

3.2.4 构建模型

3.2.5 训练模型

3.3 如何构建神经网络？

3.3.1 构建网络层

3.3.2 前向传播

3.3.3 反向传播

3.3.4 训练模型

3.4 nn.Module

3.5 nn.functional

3.6 优化器

3.7 动态修改学习率参数

3.8 优化器比较

3.9 小结

第 4 章 Pytorch 数据处理工具箱

4.1 数据处理工具箱概述

4.2 utils.data 简介

4.3 torchvision 简介

4.3.1 transforms

4.3.2 ImageFolder

4.4 可视化工具

4.4.1 tensorboardX 简介

4.4.2 用 tensorboardX 可视化神经网络

4.4.3 用 tensorboardX 可视化损失值

4.4.4 用 tensorboardX 可视化特征图

4.5 小结

第二部分 深度学习基础

第 5 章 机器学习基础

5.1 机器学习的基本任务

5.1.1 监督学习

5.1.2 无监督学习

5.1.3 半监督学习

5.1.4 强化学习

5.2 机器学习一般流程

5.2.1 明确目标

5.2.2 收集数据

5.2.3 数据探索与预处理

5.2.4 选择模型及损失函数

5.2.5 评估及优化模型

5.3 过拟合与欠拟合

5.3.1 权重正则化

5.3.2 dropout 正则化

5.3.3 批量正则化

5.3.4 权重初始化

5.4 选择合适激活函数

5.5 选择合适的损失函数

5.6 选择合适优化器

5.6.1 传统梯度优化的不足

5.6.2 动量算法

5.6.3 AdaGrad 算法

5.6.4 RMSProp 算法

5.6.5 Adam 算法

5.7 GPU 加速

5.7.1 单 GPU 加速

5.7.2 多 GPU 加速

5.7.3 使用 GPU 注意事项

5.8 小结

第 6 章 视觉处理基础

6.1 卷积神经网络简介

6.2 卷积层

- 6.2.1 卷积核
- 6.2.2 步幅
- 6.2.3 填充
- 6.2.4 多通道上的卷积
- 6.2.5 激活函数
- 6.2.6 卷积函数
- 6.2.7 转置卷积
- 6.3 池化层
 - 6.3.1 局部池化
 - 6.3.2 全局池化
- 6.4 现代经典网络
 - 6.4.1 LeNet-5 模型
 - 6.4.2 AlexNet 模型
 - 6.4.3 VGG 模型
 - 6.4.4 GoogleNet 模型
 - 6.4.5 ResNet 模型
 - 6.4.6 胶囊网络简介
- 6.5 Pytorch 实现 cifar10 多分类
 - 6.5.1 数据集说明
 - 6.5.2 加载数据
 - 6.5.3 构建网络
 - 6.5.4 训练模型
 - 6.5.5 测试模型
 - 6.5.6 采用全局平均池化
 - 6.5.7 像 keras 一样显示各层参数
- 6.6 模型集成提升性能
 - 6.6.1 使用模型
 - 6.6.2 集成方法
 - 6.6.3 集成效果
- 6.7 使用现代经典模型提升性能
- 6.8 小结

第 7 章 自然语言处理基础

- 7.1 循环神经网络基本结构
- 7.2 前向传播与随时间反向传播
- 7.3 循环神经网络变种
 - 7.3.1 LSTM
 - 7.3.2 GRU

7.3.3 Bi-RNN

7.4 循环神经网络的 Pytorch 实现

7.4.1 RNN 实现

7.4.2 LSTM 实现

7.4.3 GRU 实现

7.5 文本数据处理

7.6 词嵌入

7.6.1 Word2Vec 原理

7.6.2 CBOW 模型

7.6.3 Skip-gram 模型

7.7 Pytorch 实现词性判别

7.7.1 词性判别主要步骤

7.7.2 数据预处理

7.7.3 构建网络

7.7.4 训练网络

7.7.5 测试模型

7.8 用 LSTM 预测股票行情

7.8.1 导入数据

7.8.2 数据概览

7.8.3 预处理数据

7.8.4 定义模型

7.8.5 训练模型

7.8.6 测试模型

7.9 循环神经网络应用场景

7.10 小结

第 8 章 生成式深度学习

8.1 用变分自编码器生成图像

8.1.1 自编码器

8.1.2 变分自编码器

8.1.3 用变分自编码器生成图像

8.2 GAN 简介

8.2.1 GAN 架构

8.2.2 GAN 的损失函数

8.3 用 GAN 生成图像

8.3.1 判别器

8.3.2 生成器

8.3.3 训练模型

- 8.3.4 可视化结果
- 8.4 VAE 与 GAN 的异同
- 8.5 Condition GAN
 - 8.5.1 CGAN 的架构
 - 8.5.2 CGAN 生成器
 - 8.5.3 CGAN 判别器
 - 8.5.4 CGAN 损失函数
 - 8.5.5 CGAN 可视化
 - 8.5.6 查看指定标签的数据
 - 8.5.7 可视化损失值
- 8.6 DCGAN
- 8.7 提升 GAN 训练效果的一些技巧
- 8.8 小结
- 第三部分 深度学习实战
- 第 9 章 人脸检测与识别
 - 9.1 人脸识别一般流程
 - 9.2 人脸检测
 - 9.2.1 目标检测
 - 9.2.2 人脸定位
 - 9.2.3 人脸对齐
 - 9.2.4 MTCNN 算法
 - 9.3 特征提取
 - 9.4 人脸识别
 - 9.4.1 人脸识别主要原理
 - 9.4.2 人脸识别发展
 - 9.5 Pytorch 实现人脸检测与识别
 - 9.5.1.验证检测代码
 - 9.5.2.检测图像
 - 9.5.3.检测后进行预处理
 - 9.5.4.查看经检测后的图片
 - 9.5.5.人脸识别
 - 9.6 小结
- 第 10 章 迁移学习实例
 - 10.1 迁移学习简介
 - 10.2 特征提取
 - 10.2.1 Pytorch 提供的预处理模块
 - 10.2.2 特征提取实例

10.3 数据增强

10.3.1 按比例缩放

10.3.2 裁剪

10.3.3 翻转

10.3.4 改变颜色

10.3.5 组合多种增强方法

10.4 微调实例

10.4.1 数据预处理

10.4.2 加载预训练模型

10.4.3 修改分类器

10.4.4 选择损失函数及优化器

10.4.5 训练及验证模型

10.5 清除图像中的雾霾

10.5.1 导入需要的模块

10.5.2 查看原来的图像

10.5.3 定义一个神经网络

10.5.4 训练模型

10.5.5 查看处理后的图像

10.6 小结

第 11 章 神经网络机器翻译实例

11.1 Encoder-Decoder 模型原理

11.2 注意力框架

11.3 Pytorch 实现注意力 Decoder

11.3.1 构建 Encoder

11.3.2 构建简单 Decoder

11.3.3 构建注意力 Decoder

11.4 用注意力机制实现中英文互译

11.4.1 导入需要的模块

11.4.2 数据预处理

11.4.3 构建模型

11.4.4 训练模型

11.4.5 随机采样，对模型进行测试

11.4.6 可视化注意力

11.5 小结

第 12 章 实战生成式模型

12.1 Deep Dream 模型

12.1.1 Deep Dream 原理

- 12.1.2 DeepDream 算法流程
- 12.1.3 用 Pytorch 实现 Deep Dream
- 12.2 风格迁移
 - 12.2.1 内容损失
 - 12.2.2 风格损失
 - 12.2.3 用 Pytorch 实现神经网络风格迁移
- 12.3 Pytorch 实现图像修复
 - 12.3.1 网络结构
 - 12.3.2 损失函数
 - 12.3.3 图像修复实例
- 12.4 Pytorch 实现 DiscoGAN
 - 12.4.1 DiscoGAN 架构
 - 12.4.2 损失函数
 - 12.4.3 DiscoGAN 实现
 - 12.4.4 用 Pytorch 实现从边框生成鞋子
- 12.5 小结

第 13 章 Caffe2 模型迁移实例

- 13.1 Caffe2 简介
- 13.2 Caffe 如何升级到 Caffe2
- 13.3 Pytorch 如何迁移到 caffe2
- 13.4 小结

第 14 章 AI 新方向：对抗攻击

- 14.1 对抗攻击简介
 - 14.1.1 白盒攻击与黑盒攻击
 - 14.1.2 无目标攻击与有目标攻击
- 14.2 常见对抗样本生成方式
 - 14.2.1 快速梯度符号法
 - 14.2.2 快速梯度算法
- 14.3 Pytorch 实现对抗攻击
 - 14.3.1 实现无目标攻击
 - 14.3.2 实现有目标攻击
- 14.4 对抗攻击和防御措施
 - 14.4.1 对抗攻击
 - 14.4.2 常见防御方法分类
- 14.5 总结

第 15 章 强化学习

- 15.1 强化学习简介

15.2 Q Learning 原理

15.2.1 Q Learning 主要流程

15.2.2 Q 函数

15.2.3 贪婪策略

15.3 用 Pytorch 实现 Q Learning

15.3.1 定义 Q-Learning 主函数

15.3.2 执行 Q-Learning

15.4 SARSA 算法

15.4.1 SARSA 算法主要步骤

15.4.2 用 Pytorch 实现 SARSA 算法

15.5 小结

第 16 章 深度强化学习

16.1 DQN 算法原理

16.1.1 Q-Learning 方法的局限性

16.1.2 用 DL 处理 RL 需要解决的问题

16.1.3 用 DQN 解决方法

16.1.4 定义损失函数

16.1.5 DQN 的经验回放机制

16.1.6 目标网络

16.1.7 网络模型

16.1.8 DQN 算法

16.2 用 Pytorch 实现 DQN 算法

16.3 小结

附录 A:Pytorch0.4 版本变更

A.1 概述

A.2 合并 Variable 和 Tensor

A.3 弃用 volatile 标签

A.4 dypes,devices 以及 numpy-style 的构造函数

A.5 迁移实例比较

附录 B:AI 在各行业的最新应用

B.1 AI+电商

B.2 AI+金融

B.3 AI+医疗

B.4 AI+零售

B.5 AI+投行

B.6 AI+制造

B.7 AI+IT 服务

B.8 AI+汽车

B.9 AI+公共安全