

第一天

2019年6月6日 9:27

1. 深度学习的概念

- a. 机器学习的分支，以神经网络为基础，对数据的特征进行学习。

2. 和机器学习的区别

- a. 深度学习不需要手动的进行特征工程
- b. 深度学习需要的数量多，需要的计算性能强

深度学习的应用场景

常见框架

神经网络

1. 概念

- a. 模仿生物的神经系统实现的模型，能够对数据的特征进行学习

2. 神经元

- a. 神经网络中的最小的单元。不同的神经元组合能够得到神经网络
- b. 结构： $t = f(Wx + b)$
- c. 内积：点积，向量的乘法（对应位置相乘相加），得到一个标量。

3. 单层的神经网络

4. 两层的神经网络：

- a. 感知机：两层神经网络。输入层有多个神经元，输出层一个神经元。
- b. 感知机的作用：进行一个二分类

5. 多层神经网络：

- a. 每一层的神经元之间没有连接
- b. 全连接层：当N层的每一个神经元和前一层的每一个神经元都有链接的时候，第N层就是全连接层。就是在进行矩阵乘法，进行特征的变换，就在进行 $y = wx + b$ 。

6. 激活函数：把原来的数据进行变换

- a. 为什么要使用非线性的激活函数

- i. 线性的激活函数或者是不是用激活函数，多层神经网络和两层神经网络没有区别
- ii. 是用非线性的激活函数能够增加模型的非线性分割能力

- b. 为什么要使用简单的激活函数（RELU）

- i. RELU方便求导
- ii. sigmoid在取值很大或者很小的时候，导数非常小，会导致参数的更新速度很慢

- c. 激活函数的作用:
 - i. 增加非线性分割能力
 - ii. 增加模型的稳健性 (让模型能够拟合不同的数据)
 - iii. 缓解梯度消失
 - iv. 加速模型收敛

pytorch的使用

- 1. 张量
 - a. 0阶张量: 常数, eg: 1
 - b. 1阶张量: 向量, eg: [1,2,3]
 - c. 2阶张量: 矩阵, eg: [[1,2]]
- 2. 张量的创建
 - a. `torch.tensor(list/array)`
 - b. `torch.ones/empty/zeros([3,4])`
 - c. `torch.rand([3,4]) , [0,1)`
 - d. `torch.randint(low,high,size=[])`: 生成size个取值从low到high的随机整数
 - e. `torch.randn([3,4])` 均值为0标准差为1的数组
- 3. 张量相关的属性和方法
 - a. tensor中只有一个元素的时候, 获取数据
 - i. `tensor.item()`
 - b. tensor转化为ndarray
 - i. `tensor.numpy()`
 - c. 获取tensor的形状
 - i. `tensor.size(dim)`
 - d. tensor的变形和转置
 - i. `tensor.view()`
 - ii. `tensor.t(0,1)`
 - e. tensor的切片和索引
 - i. 和numpy相同
 - f. 获取tensor中的最大值
 - i. `tensor.max()`
 - ii. `tensor.max(dim=-1) #dim=-1获取行方向的最大值`
- 4. tensor的数据类型和修改方法
 - a. `int32 --> int`
 - b. `int64 --> long`
 - c. `float32-->float`

- d. float64 ---> double
 - e. tensor.float().long().double().int()
5. tensor的计算
- a. tensor+tensor, 形状相同对应位置计算
 - b. 形状不同: 可以进行广播
 - c. tensor+数字 ,tensor中的每个值和数字计算
 - d. x.add_(y) x的值会直接被修改
6. CUDA类型的tensor
- a. device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
 - b. torch.tensor([1,,3],device) #创建cuda类型的tensor
 - c. tensor.to(device) #把tensor转化为cuda支持 的tensor
 - d. tensor.cpu() #把cuda的tensor转化为cpu上的tensor