第一次打卡学习笔记

Task 1

基础理论

深度学习的基础是要求有数据，数据从收集的一批关于想要学习的的东西而来，比如有关于房价和平方的相互关系。这组集合就叫做为数据集，每条记录关于一个对像或事件的描述，称为样本，反映了在一个方面的表现或性质。样本中的每个描述有属性和值，这些组合在一起就形成了一个数据集，从数据集中学习或者是训练，学习过程是通过执行不同的算法来完成的。

深度学习主要的两个方面是进行离散值的预测，例如好坏，这个称为是分类（Classification）；若预测的值为连续值，例如线性的一个公式这类就是称为回归。

学习得到模型后还要进行测试，使用测试的用例来进行预测，还有一个就是聚类，将训练集中的数据分为若干组，每个组叫做一个簇。这些自动形成的簇，会对应一些潜在的概念划分，这种学习能够了解数据集内在的规律。

线性模型

Softmax回归模型

多层感知机

有多层感知机的隐藏层也只是造价于一个单层神经网络。输出的权重参数为

偏差参数为

三种方式的比较

|  |  |  |  |
| --- | --- | --- | --- |
|  | 线性回归 | Softmax回归 | 多层感知机 |
| 学习任务 | 回归 | 分类 | 分类 |
| 神经网络层数 | 单层神经网络 | 单层神经网络 | 多层神经网络 |
| 隐藏层 | 无 | 无 | 有 |
| 实现过程 | 1. 生成数据集 2. 初始化模型参数 3. 定义模型 4. 定义损失函数 5. 定义优化算法 6. 训练模型 | 1. 获取读取数据 2. 初始化模型参数 3. 进行softmax运算 4. 定义模型 5. 定义损失函数 6. 计算分类准确率 7. 训练模型 | 1. 获取和读取数据 2. 初始化模型参数 3. 定义激活函数 4. 定义模型 5. 定义损失函数 6. 训练模型 |
| 激活函数（作用是在超过一个阈值后进行，激活使用） | \ | \ | ReLu（rectified linear unit）非线性变换只保留正元素负元素清零  Sigmoid值在0至1间  Tanh(双曲正切)将元素值变换到-1至1间 |

Pytorch

Torch.tensor()

是一种包含单一数据类型元素的多维矩阵, 是默认的tensor类型（torch.FlaotTensor）的简称。

torch.randn

torch.randn(\*sizes, out=None) → Tensor

返回一个张量，包含了从标准正态分布(均值为0，方差为 1，即高斯白噪声)中抽取一组随机数，形状由可变参数sizes定义。 参数:

sizes (int...) – 整数序列，定义了输出形状

out (Tensor, optinal) - 结果张量

Torch.zeros()

torch.zeros(\*sizes, out=None) → Tensor

返回一个全为标量 0 的张量，形状由可变参数sizes 定义。

sizes (int...) – 整数序列，定义了输出形状

out (Tensor, optional) – 结果张量

Torch.optim()

是一个实现了各种优化算法的库。大部分常用的方法得到支持，并且接口具备足够的通用性，使得未来能够集成更加复杂的方法。为了使用torch.optim，你需要构建一个optimizer对象。这个对象能够保持当前参数状态并基于计算得到的梯度进行参数更新。

为了构建一个Optimizer，你需要给它一个包含了需要优化的参数（必须都是Variable对象）的iterable。然后，你可以设置optimizer的参 数选项，比如学习率，权重衰减，等等。

例子：

optimizer = optim.SGD(model.parameters(), lr = 0.01, momentum=0.9)

optimizer = optim.Adam([var1, var2], lr = 0.0001)

(params, lr=, momentum=0, dampening=0, weight\_decay=0, nesterov=False)

params (iterable) – 待优化参数的iterable或者是定义了参数组的dict

lr (float) – 学习率

momentum (float, 可选) – 动量因子（默认：0）

weight\_decay (float, 可选) – 权重衰减（L2惩罚）（默认：0）

dampening (float, 可选) – 动量的抑制因子（默认：0）

nesterov (bool, 可选) – 使用Nesterov动量（默认：False）

torch.mm()

torch.mm(mat1, mat2, out=None) → Tensor

对矩阵mat1和mat2进行相乘。 如果mat1 是一个n×m 张量，mat2 是一个 m×p 张量，将会输出一个 n×p 张量out。

参数 ：

mat1 (Tensor) – 第一个相乘矩阵

mat2 (Tensor) – 第二个相乘矩阵

out (Tensor, optional) – 输出张量

torchvision包 包含了目前流行的数据集，模型结构和常用的图片转换工具

dset.MNIST(root, train=True, transform=None, target\_transform=None, download=False)

参数说明： - root : processed/training.pt 和 processed/test.pt 的主目录 - train : True = 训练集, False = 测试集 - download : True = 从互联网上下载数据集，并把数据集放在root目录下. 如果数据集之前下载过，将处理过的数据（minist.py中有相关函数）放在processed文件夹下。

信息处理

英文文本词频统计

1. 正则表达式

re.sub (﻿'[^a-z]+'﻿, ' '﻿, str﻿)*﻿*

re.sub()函数是用来字符串替换的函数﻿, 将字符串str中的非小写字母开头的字符串以空格代替

string = ' AbcDe' # 主要是说明string是一个字符串,下面是视频中涉及到了两个操作

string.strip(﻿) # 用于移除字符串头尾指定的字符（默认为空格或换行符）或字符序列，在这里是用来删除单词之间的空格和换行符

string.lower(﻿) # 将字符串中的所有大写字母都转化成小写﻿

1. 分词
2. 字典

建立索引，建立类，划分词类

1. 采样判断

都是基于概率的要求

由1阶马尔科夫链，从第二个词开始每个词只与其前一个词有关。

由2阶马尔科夫链，从第三个词开始每个词只与其前两个词有关。

由3阶马尔科夫链，从第四个词开始每个词只与其前三个词有关。

给定训练数据[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]，批量大小为batch\_size=2，时间步数为2，使用本节课的实现方法进行采样，第二个批量为：

随机采样：因为训练数据中总共有11个样本，而批量大小为2，所以数据集会被拆分成2段，每段包含5个样本。

相邻采样：因为训练数据中总共有11个样本，而批量大小为2，所以数据集会被拆分成2段，每段包含5个样本：[0, 1, 2, 3, 4]和[5, 6, 7, 8, 9]，而时间步数为2，所以第二个批量为[2, 3]和[7, 8]。