深度学习笔记

Task0

一、 线性回归

1、线性回归根据输入数据预测输出，输入与输出呈线性关系

2、深度模型的基本要素包括:

数据集（训练集、验证集、测试集）

损失函数

优化函数

训练模型

深度模型训练的基本步骤:

读取数据

设置超参数

初始化模型参数

前向传播计算输出

计算 loss

反向传播计算梯度

优化器更新参数

二、 Softmax 与分类模型

1、softmax 计算

$$\hat{y}_1, \hat{y}_2, \hat{y}_3 = \text{softmax}(o_1, o_2, o_3)$$

$$\hat{y}1 = \frac{\exp(o_1)}{\sum_{i=1}^{3} \exp(o_i)}, \quad \hat{y}2 = \frac{\exp(o_2)}{\sum_{i=1}^{3} \exp(o_i)}, \quad \hat{y}3 = \frac{\exp(o_3)}{\sum_{i=1}^{3} \exp(o_i)}.$$

2、交叉熵计算

$$H\left(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}\right) = -\sum_{j=1}^{q} y_j^{(i)} \log \hat{y}_j^{(i)},$$

三、 多层感知机

1、多层感知机为至少含有一层隐藏层的多层网络，如果全连接层之间不加激活函数层的话，则多个线性层叠加与只有一层的表达能力相同。激活函数对线性层的输出进行非线性化

2、各种激活函数

Relu、sigmoid、tanh

关于激活函数的选择

ReLu函数是一个通用的激活函数，目前在大多数情况下使用。但是，ReLU函数只能在隐藏层中使用。

用于分类器时，sigmoid函数及其组合通常效果更好。由于梯度消失问题，有时要避免使用sigmoid和tanh函数。

在神经网络层数较多的时候，最好使用ReLu函数，ReLu函数比较简单计算量少，而sigmoid和tanh函数计算量大很多。

在选择激活函数的时候可以先选用ReLu函数如果效果不理想可以尝试其他激活函数。

Task1

一、 文本预处理

文本预处理基本步骤：

1、 读入文本

2、 进行特定级别的分词

3、 建立字典，为每个词映射到一个唯一的索引

4、 将文本从词的序列转为索引的序列，方便模型输出

二、 语言模型

1、 语言模型的目的是为了验证一段词的序列是否合理

假设序列$w_1, w_2, \ldots, w_T$中的每个词是依次生成的，我们有

$$\begin{aligned} P(w_1, w_2, \ldots, w_T) &= \prod_{t-1}^{T} P(w_t \mid w_1, \ldots, w_{t-1}) \\ &= P(w_1)P(w_2 \mid w_1)\cdots P(w_T \mid w_1 w_2 \cdots w_{T-1}) \end{aligned}$$

2、 n 元语法

序列长度增加，计算和存储多个词共同出现的概率的复杂度会呈指数级增加。n元语法通过马尔可夫假设简化模型，马尔科夫假设是指一个词的出现只与前面n个词相关，即n阶马尔可夫链（Markov chain of order n），如果$n=1$，那么有$P(w_3 \mid w_1, w_2) = P(w_3 \mid w_2)$。基于$n-1$阶马尔可夫链，我们可以将语言模型改写为

$$P(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_{t-(n-1)}, \ldots, w_{t-1}).$$

以上也叫n元语法（n-grams），它是基于$n-1$阶马尔可夫链的概率语言模型。例如，当$n=2$时，含有4个词的文本序列的概率就可以改写为：

$$\begin{aligned} P(w_1, w_2, w_3, w_4) &= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2)P(w_4 \mid w_1, w_2, w_3) \\ &= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_2)P(w_4 \mid w_3) \end{aligned}$$

当n分别为1、2和3时，我们将其分别称作一元语法（unigram）、二元语法（bigram）和三元语法（trigram）。例如，长度为4的序列w_1, w_2, w_3, w_4在一元语法、二元语法和三元语法中的概率分别为

$$\begin{aligned} P(w_1, w_2, w_3, w_4) &= P(w_1)P(w_2)P(w_3)P(w_4), \\ P(w_1, w_2, w_3, w_4) &= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_2)P(w_4 \mid w_3), \\ P(w_1, w_2, w_3, w_4) &= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2)P(w_4 \mid w_2, w_3). \end{aligned}$$

当n较小时，n元语法往往并不准确。例如，在一元语法中，由三个词组成的句子“你走先”和“你先走”的概率是一样的。然而，当n较大时，n元语法需要计算并存储大量的词频和多词相邻频率。

n 元语法存在的问题：

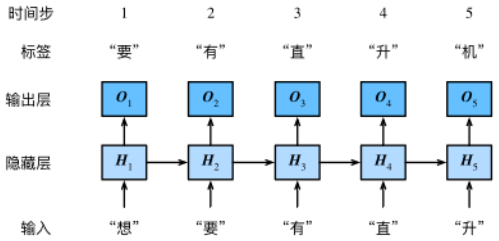
参数空间过大；
数据稀疏，因为很多词的词频太小导致出现的概率太小，很多概率接近 0

3、 时序数据采样
随机采样
相邻采样

三、 循环神经网络基础

1、 概念

本节介绍循环神经网络，下图展示了如何基于循环神经网络实现语言模型。我们的目的是基于当前的输入与过去的输入序列，预测序列的下一个字符。循环神经网络引入一个隐藏变量H，用H_t表示H在时间步t的值。H_t的计算基于X_t和H_{t-1}，可以认为H_t记录了到当前字符为止的序列信息，利用H_t对序列的下一个字符进行预测。



2、one-hot 向量

我们需要将字符表示成向量，这里采用one-hot向量。假设词典大小是N，每次字符对应一个从0到$N-1$的唯一的索引，则该字符的向量是一个长度为N的向量，若字符的索引是i，则该向量的第i个位置为1，其他位置为0。下面分别展示了索引为0和2的one-hot向量，向量长度等于词典大小。

3、梯度裁剪

由于循环神经网络容易出现梯度消失和梯度爆炸的问题，为应对梯度爆炸的问题，采用梯度裁剪的方法

循环神经网络中较容易出现梯度衰减或梯度爆炸，这会导致网络几乎无法训练。裁剪梯度（clip gradient）是一种应对梯度爆炸的方法。假设我们把所有模型参数的梯度拼接成一个向量$\boldsymbol{g}$，并设裁剪的阈值是θ。裁剪后的梯度

$$\min\left(\frac{\theta}{\|\boldsymbol{g}\|}, 1\right)\boldsymbol{g}$$

的L_2范数不超过θ。

4、困惑度

我们通常使用困惑度（perplexity）来评价语言模型的好坏。回忆一下“softmax回归”一节中交叉熵损失函数的定义。困惑度是对交叉熵损失函数做指数运算后得到的值。特别地，

- 最佳情况下，模型总是把标签类别的概率预测为1，此时困惑度为1；
- 最坏情况下，模型总是把标签类别的概率预测为0，此时困惑度为正无穷；
- 基线情况下，模型总是预测所有类别的概率都相同，此时困惑度为类别个数。

显然，任何一个有效模型的困惑度必须小于类别个数。在本例中，困惑度必须小于词典大小vocab_size。

5、模型训练

跟之前章节的模型训练函数相比，这里的模型训练函数有以下几点不同：

1. 使用困惑度评价模型。
2. 在迭代模型参数前裁剪梯度。
3. 对时序数据采用不同采样方法将导致隐藏状态初始化的不同。

对于随机采样，要在每个 batch 开始时对隐藏层参数初始化，因为相邻的两个 batch 的数据不是相邻的，所以前一个 batch 的隐藏状态不能用于下一个 batch

对于相邻采样，由于相邻的两个 batch 的数据也是相邻的，因此只需要在每个 epoch 开始时对隐藏层参数进行初始化即可