实验三：循环神经网络

姓名：刘威

学号：PB18010469

实验目的

- 了解并熟悉循环神经网络的原理
- 了解随时间反向传播算法（BPTT）
- 学会使用循环神经网络完成文本分类任务

实验原理

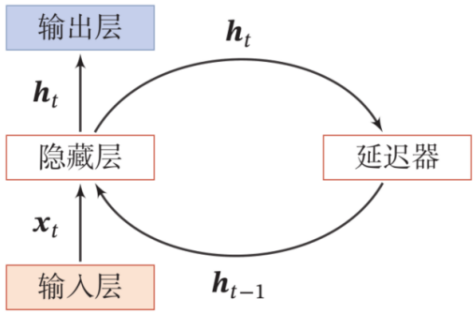
循环神经网络（Recurrent Neural Network）

- 循环神经网络通过使用带自反馈的神经元，能够处理任意长度的时序数据。

$$\boldsymbol{h}_t = f_W(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$$

新状态　　旧状态　　t时刻的输入向量

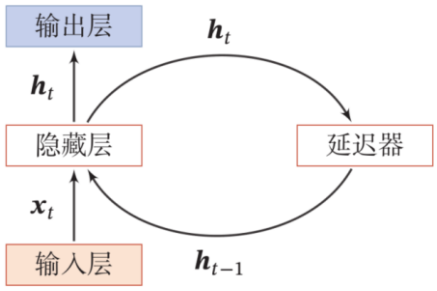
参数为W的函数



状态 $\boldsymbol{h}_t$ “存储”直到t时刻的历史数据$\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t\}$

$$\boldsymbol{h}_t = f_W(f_W(f_W(\boldsymbol{h}_{t-3}, \boldsymbol{x}_{t-2}), \boldsymbol{x}_{t-1}), \boldsymbol{x}_t)$$

Elman RNN



$$\boldsymbol{h}_t = f_{\boldsymbol{W}}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$$

$$\boldsymbol{h}_t = f(\boldsymbol{U}\boldsymbol{h}_{t-1} + \boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{b})$$

$\boldsymbol{U}$为状态-状态权重矩阵，$\boldsymbol{W}$为状态-输入权重矩阵，$\boldsymbol{b}$为偏置向量

循环神经网络的计算能力

- 一个完全连接的循环网络

$$\boldsymbol{h}_t = f(\boldsymbol{U}\boldsymbol{h}_{t-1} + \boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{b})$$
$$\boldsymbol{y}_t = \boldsymbol{V}\boldsymbol{h}_t$$

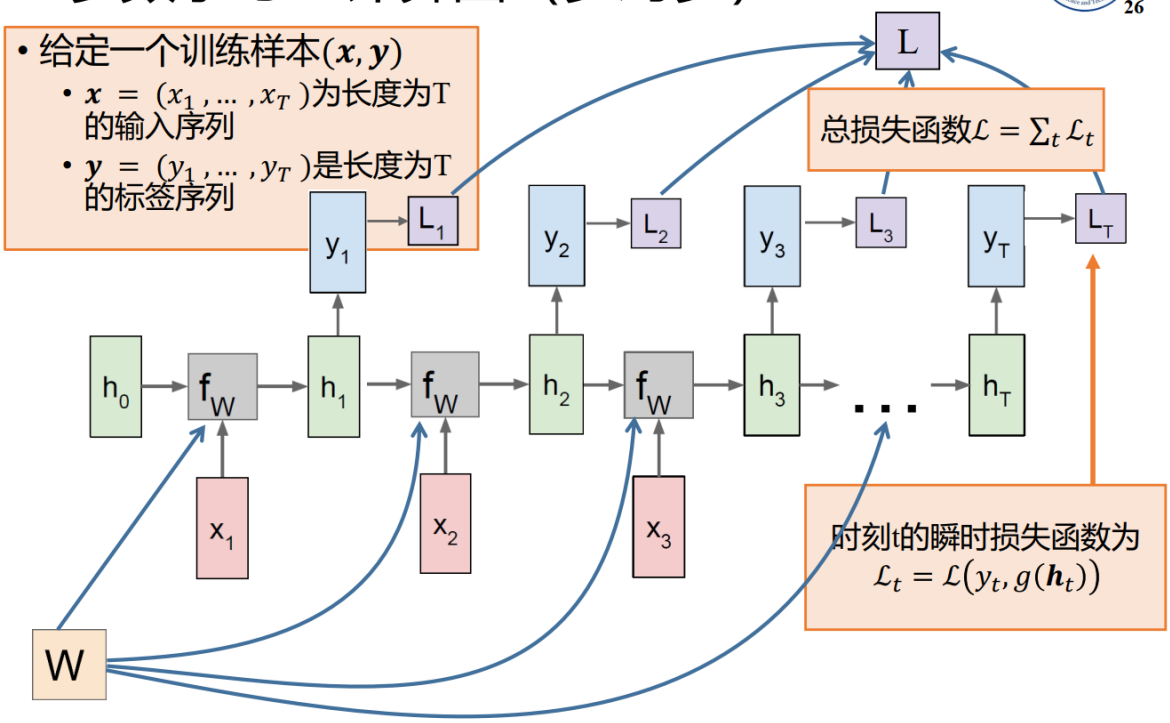
则该网络是任何非线性动力系统的近似器

定理 6.1 – 循环神经网络的通用近似定理 [Haykin, 2009]：如果一个完全连接的循环神经网络有足够数量的 sigmoid 型隐藏神经元，它可以以任意的准确率去近似任何一个非线性动力系统

$$\boldsymbol{s}_t = g(\boldsymbol{s}_{t-1}, \boldsymbol{x}_t), \tag{6.10}$$
$$\boldsymbol{y}_t = o(\boldsymbol{s}_t), \tag{6.11}$$

其中 $\boldsymbol{s}_t$ 为每个时刻的隐状态，$\boldsymbol{x}_t$ 是外部输入，$g(\cdot)$ 是可测的状态转换函数，$o(\cdot)$ 是连续输出函数，并且对状态空间的紧致性没有限制.

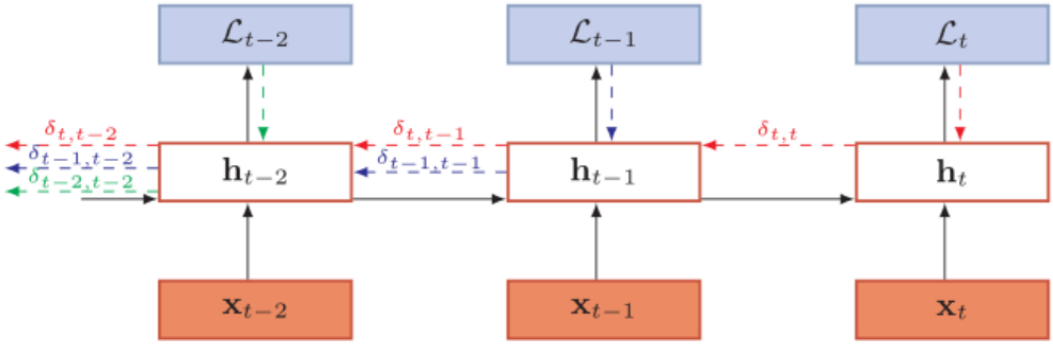
参数学习—计算图（多对多）



梯度计算

• 随时间反向传播算法（BPTT）

$$\boldsymbol{h}_{t+1} = f(\boldsymbol{z}_{t+1}) = f(\boldsymbol{U}\boldsymbol{h}_t + \boldsymbol{W}\boldsymbol{x}_{t+1} + \boldsymbol{b})$$



$\delta_{t,k}$为第t时刻的损失对第k步隐藏神经元的净输入$\boldsymbol{z}_k$的导数

$$\boldsymbol{\delta}_{t,k} = \frac{\partial \mathcal{L}_t}{\partial \boldsymbol{z}_k} = \frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{z}_k} \frac{\partial \boldsymbol{z}_{k+1}}{\partial \boldsymbol{h}_k} \frac{\partial \mathcal{L}_t}{\partial \boldsymbol{z}_{k+1}}$$

$$= \mathrm{diag}\big(f'(\boldsymbol{z}_k)\big)\, \boldsymbol{U}^{\mathrm{T}}\, \boldsymbol{\delta}_{t,k+1}$$

梯度计算

• 随时间反向传播算法（BPTT） $\boldsymbol{h}_{t+1} = f(\boldsymbol{z}_{t+1}) = f(\boldsymbol{U}\boldsymbol{h}_t + \boldsymbol{W}\boldsymbol{x}_{t+1} + \boldsymbol{b})$

$$\boldsymbol{\delta}_{t,k} = \frac{\partial \mathcal{L}_t}{\partial \boldsymbol{z}_k} = \mathrm{diag}\big(f'(\boldsymbol{z}_k)\big)\boldsymbol{U}^\top \boldsymbol{\delta}_{t,k+1}$$

$$\boldsymbol{z}_k = \boldsymbol{U}\boldsymbol{h}_{k-1} + \boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{b}$$

$$\frac{\partial \mathcal{L}_t}{\partial u_{ij}} = \sum_k^t \frac{\partial \boldsymbol{z}_k}{\partial u_{ij}} \frac{\partial \mathcal{L}_t}{\partial \boldsymbol{z}_k} = \sum_{k=1}^t \sum_l \frac{\partial z_{k,l}}{\partial u_{ij}} \frac{\partial \mathcal{L}_t}{\partial z_{k,l}}$$

$$= \sum_{k=1}^t \sum_l \mathbb{I}(l=i)[\boldsymbol{h}_{k-1}]_j \big[\boldsymbol{\delta}_{t,k}\big]_l \quad \Rightarrow \quad \frac{\partial \mathcal{L}_t}{\partial \boldsymbol{U}} = \sum_{k=1}^t \boldsymbol{\delta}_{t,k}\boldsymbol{h}_{k-1}^\top$$

$$= \sum_{k=1}^t [\boldsymbol{h}_{k-1}]_j \big[\boldsymbol{\delta}_{t,k}\big]_i$$

梯度计算

• 随时间反向传播算法（BPTT） $\boldsymbol{h}_{t+1} = f(\boldsymbol{z}_{t+1}) = f(\boldsymbol{U}\boldsymbol{h}_t + \boldsymbol{W}\boldsymbol{x}_{t+1} + \boldsymbol{b})$

$$\boldsymbol{\delta}_{t,k} = \frac{\partial \mathcal{L}_t}{\partial \boldsymbol{z}_k} = \mathrm{diag}\big(f'(\boldsymbol{z}_k)\big)\boldsymbol{U}^\top \boldsymbol{\delta}_{t,k+1} = \prod_{\tau=k}^{t-1} \big(\mathrm{diag}\big(f'(\boldsymbol{z}_\tau)\big)\boldsymbol{U}^\top\big)\, \boldsymbol{\delta}_{t,t}$$

$\|\boldsymbol{U}\| = r_1$

如果$|f'(\boldsymbol{z}_\tau)| \le \gamma$，
$\boldsymbol{U}$的最大奇异值$\lambda_1 < \frac{1}{\gamma}$ $\Rightarrow$ $\big\|\mathrm{diag}\big(f'(\boldsymbol{z}_\tau)\big)\boldsymbol{U}^\top\big\| \le \gamma\|\boldsymbol{U}\| < 1$

$|\tanh(x)| \le 1$
$|\sigma(x)| \le \frac{1}{4}$

一定存在$\eta < 1$，使得$\big\|\mathrm{diag}\big(f'(\boldsymbol{z}_\tau)\big)\boldsymbol{U}^\top\big\| \le \eta$

$$\Rightarrow \quad \|\boldsymbol{\delta}_{t,k}\| \le \left\| \prod_{\tau=k}^{t-1} \big(\mathrm{diag}\big(f'(\boldsymbol{z}_\tau)\big)\boldsymbol{U}^\top\big)\, \boldsymbol{\delta}_{t,t} \right\| \le \eta^{t-k}\|\boldsymbol{\delta}_{t,t}\|$$

$t-k \to \infty$
$\Rightarrow$ $\|\boldsymbol{\delta}_{t,k}\| \to 0$

长程依赖问题

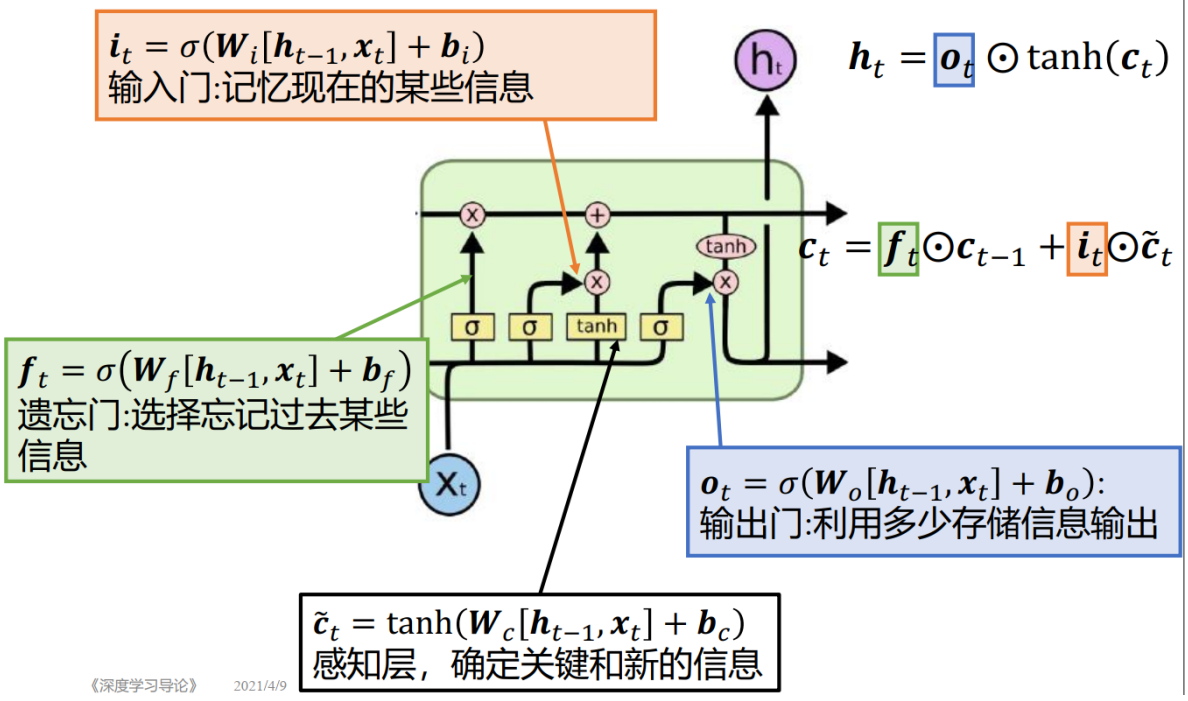
- 由于梯度爆炸或消失问题，实际上只能学习到短周期的依赖关系。这就是所谓的长程依赖问题

- 循环神经网络中的梯度消失不是$\frac{\partial \mathcal{L}_t}{\partial \boldsymbol{U}}$的梯度消失，而是$\frac{\partial \mathcal{L}_t}{\partial \boldsymbol{h}_k}$的梯度消失

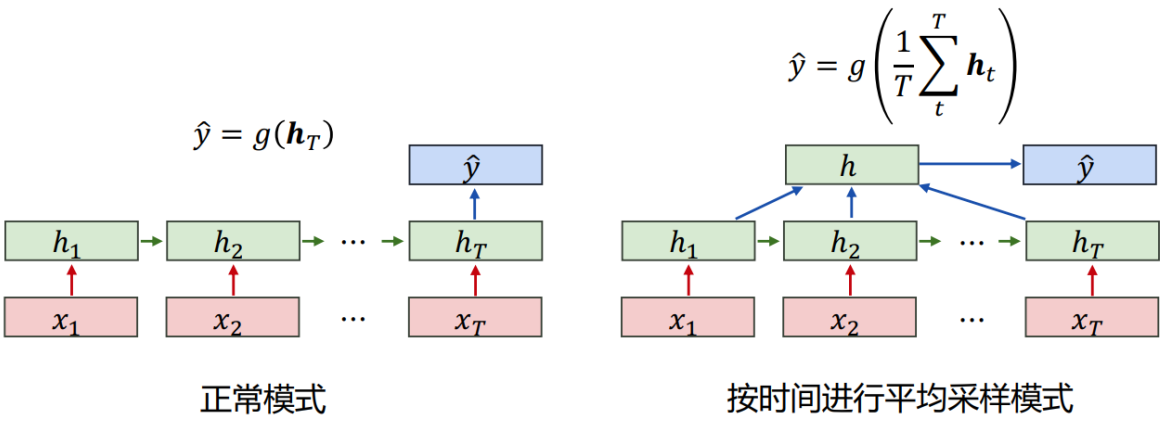
$$\frac{\partial \mathcal{L}_t}{\partial \boldsymbol{h}_k} = \boldsymbol{U}^{\top} \operatorname{diag}\left(f'(\boldsymbol{z}_{k+1})\right) \frac{\partial \mathcal{L}_t}{\partial \boldsymbol{h}_{k+1}}$$

- U的更新主要靠当前时刻t的几个相邻状态$\boldsymbol{h}_k$来更新，长距离的状态对参数几乎没有影响

长短期记忆神经网络（LSTM）



序列到类别



正常模式　　按时间进行平均采样模式

$g(\cdot)$可以是简单的线性分类器（对率回归）或者复杂分类器（多层前馈神经网络）

实验内容

编写RNN的语言模型，并基于训练好的词向量，编写RNN模型用于文本分类

数据集：aclIMDB

预训练词向量：GloVe.6B

实验结果

实验使用`pytorch`进行

源码结构及说明

数据预处理部分

使用`torchtext`库处理文本；使用`spaCy`库进行分词。

将`train/`目录下的数据集划分为`train/validation`，划分比例为`0.8/0.2`.

模型部分

由一个`Embedding`层和一个`RNN/LSTM`模块构成，后者可以调节层数和是否双向。

`Embedding`层使用`GloVe`预训练词向量进行初始化。

模型定义如下：

```
class RNNClassifier(nn.Module):
    def __init__(self, vocab_size, embedding_dim, hidden_dim,
                 n_layers: int = 1, bidirectional: bool = False,
                 dropout: float = 0., model_base: str = 'RNN'):
        super(RNNClassifier, self).__init__()
        self.bidirectional = bidirectional
        self.model_base = model_base.lower()
        if self.model_base == 'lstm':
            model = nn.LSTM
        else:
```

```
            model = nn.RNN

        self.embedding = nn.Embedding(vocab_size, embedding_dim)
        self.rnn = model(embedding_dim,
                         hidden_dim,
                         num_layers=n_layers,
                         bidirectional=bidirectional,
                         dropout=dropout)
        if self.bidirectional:
            hidden_dim *= 2
        self.fc = nn.Linear(hidden_dim, 1)
        self.act = nn.Sigmoid()

    def forward(self, x, x_len):
        x = self.embedding(x)
        x = pack_padded_sequence(x, x_len)
        if self.model_base == 'lstm':
            _, (h_n, _) = self.rnn(x)
        else:
            _, h_n = self.rnn(x)  # h_n.shape = (num_layers * num_directions,
batch, hidden_size)
        if self.bidirectional:
            hidden = torch.cat((h_n[-2], h_n[-1]), dim=1)  # get last layer
        else:
            hidden = h_n[-1]
        logits = self.fc(hidden)
        output = self.act(logits)
        return output
```

结果及分析

本实验的可选参数为

```
VOCAB_SIZE = 400000
EMBEDDING_DIM = 100
HIDDEN_DIM = 64
N_LAYERS = 1  # RNN/LSTM 层数
BIDIRECTIONAL = False  # 是否双向
DROPOUT = 0.
BATCH_SIZE = 128
N_EPOCHS = 10
MODEL_BASE = 'RNN'  # 使用`Elman RNN` 还是 `LSTM`
```

此外，本实验固定随机种子：

```
import torch
import random
import os
import numpy as np

def set_seed(seed=123):
    random.seed(seed)
    np.random.seed(seed)
    os.environ["PYTHONHASHSEED"] = str(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
```

```
    # torch.use_deterministic_algorithms(True)
    # torch.backends.cudnn.enabled = False
    torch.backends.cudnn.benchmark = False
    torch.backends.cudnn.deterministic = True
    os.environ["CUBLAS_WORKSPACE_CONFIG"] = ":4096:2"

set_seed(2077)
```

词向量维度：100维

在验证集上验证保存`val_loss`最低的模型用于测试，得到的测试集准确率(%)如下表:

N_L-N_D	RNN	LSTM
1-1	77.22	85.57
1-2	77.87	85.64
2-1	77.30	86.56
2-2	76.41	85.65
5-1	75.28	84.47
5-2	74.14	84.56

注： `N_L`代表`N_Layers`，及循环神经网络的层数， `N_D`代表`N_Direction`，当`bidirectional`设为`False`时为`1`， 否则为`2`.

结果表明，`LSTM`明显优于`RNN`；设置双向对于网络浅时略有提升，对于网络深时有副作用；简单地加深网络会使模型性能变差。

词向量维度：300维

将词向量维度增加到300维，比较模型表现。

N_L-N_D	RNN	LSTM
1-1	76.29	86.20
1-2	76.80	86.62

RNN性能变差，LSTM性能变好。

实验总结

本实验地主要难点在于：

- 认清并理解完成任务所需要地流程；
- 文本处理的流程。
- 词向量嵌入的原理和实践方法。

因此主要时间花在如何处理数据上。模型结构上相比前几次实验反而要简单一些。