南开大学
Nankai University

南　　开　　大　　学

网 络 空 间 安 全 学 院

深度学习平时作业

RNN 实验报告

穆禹宸　2012026

年级：2020 级

专业：信息安全、法学双学位班

指导教师：侯淇彬

2023 年 6 月 23 日

目录

一、 实验过程

(一) 原始版本 RNN

1. 网络结构

```
RNN(
  (i2h): Linear(in_features=185, out_features=128, bias=True)
  (i2o): Linear(in_features=185, out_features=18, bias=True)
  (softmax): LogSoftmax(dim=1)
)
```

2. 实验结果

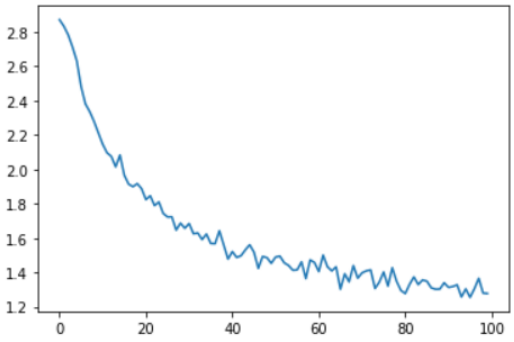

图 1: **原始 RNN 的损失**

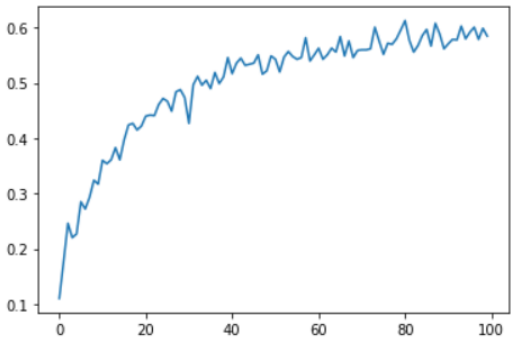

图 2: **原始 RNN 的准确率**

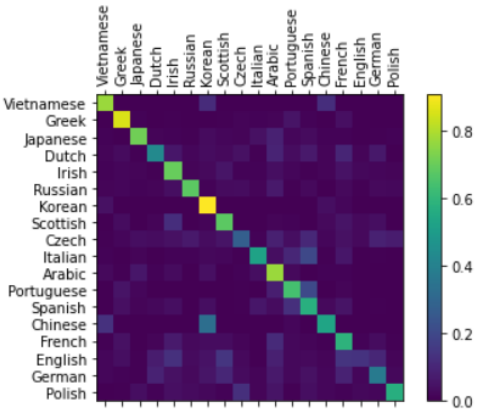

图 3: **原始 RNN 的预测矩阵图**

(二) LSTM

1. 网络结构

```
LSTM(
  (rnn): LSTM(57, 128)
  (out): Linear(in_features=128, out_features=18, bias=True)
  (softmax): LogSoftmax(dim=1)
)
```

这里仅仅实现了一个简单的 LSTM。

2. 实验结果

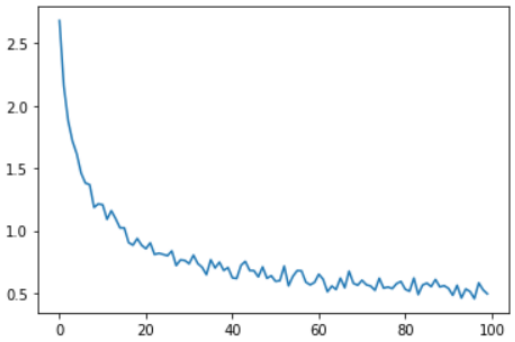

图 4: **LSTM 的损失**

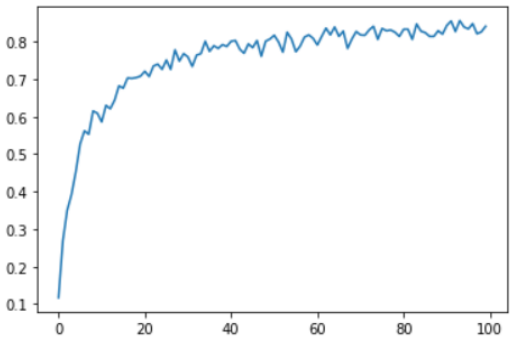

图 5: **LSTM 的准确率**

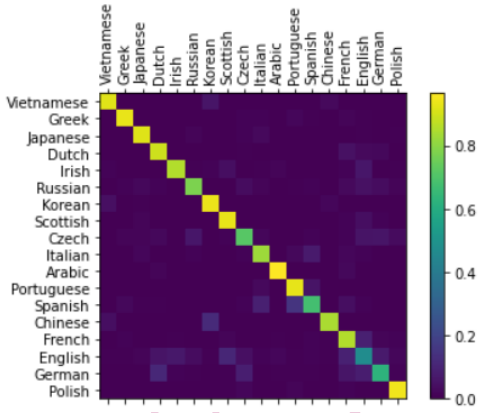

图 6: **LSTM 的预测矩阵图**

二、 解释为什么 LSTM 网络的性能优于 RNN 网络

下面我们先解释一下 RNN 的结构。

(一) RNN

RNN 将当前时间步的输入 x_t 和上个时间步的隐藏状态 h_{t-1} 经过一个线性变换和一个非线性激活函数 g 得到当前时间步的隐藏状态 h_t，如公式中的 a_t 所示:

$$a_t = W_{aa}h_{t-1} + W_{ax}x_t + b_a$$
$$h_t = g(a_t)$$

其中，W_{aa}、W_{ax} 和 b_a 是可学习的参数。

然而，这种结构并不能处理很长的序列，因为 RNN 的信息传递是链式的，每个时间步对前面的状态进行一次线性变换，因此，时间步 t 的状态 h_t 包含了历史时刻的信息，也包括当前时刻的信息，但是历史信息通常需要对 $\sum_{i=t-k}^{t} h_i$ 求和，其中 k 是一个比较大的值，而这个和有时候可能没有特别明显的贡献。

(二) LSTM

LSTM 则采用了门控机制，每个时间步维护三个门以及一个新的记忆单元。其中，遗忘门控制了上个时刻记忆单元中什么信息将被保留，输入门确定哪些信息将进入记忆单元，输出门用于

生成当前时刻的输出。具体来说，对于时刻 t，LSTM 的计算公式为:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$\widetilde{C_t} = tanh(W_c x_t + U_c h_{t-1} + b_c)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C_t}$$

$$h_t = o_t \odot tanh(C_t)$$

其中，f_t、i_t 和 o_t 分别为遗忘门、输入门和输出门，这些门通常是由 Sigmoid 函数来实现的，而 $\widetilde{C_t}$ 是候选记忆单元，即这是我们可能选择加入当前时刻的记忆单元。然后，记忆单元状态 C_t 是由上一时刻记忆单元状态 C_{t-1} 和当前时刻候选记忆单元状态 $\widetilde{C_t}$ 一起运算得到的。最后，输出 h_t 是由记忆单元状态 C_t 计算得到的，这样，LSTM 不仅可以保留当前状态，同时也维护了历史状态中需要保留的信息。

可以看出，LSTM 的计算公式比 RNN 的公式复杂得多，但是这种复杂性允许它能够在长序列中捕获长期依赖关系。相比于传统的 RNN，增加了三个门控机制（输入门/遗忘门/输出门）以及细胞状态 C_t，这些机制使得 LSTM 在对序列数据进行建模时的表现能力更强。

因此，核心在于，自然语言这种长序列的数据形式之中的梯度消失问题和梯度爆炸问题，是由于在梯度反向传播过程中的链式乘积导致了梯度值的不稳定。在 RNN 中，梯度能够一直传递到第一个时间步，并且随着时间的增加而指数级增长或减小，导致梯度爆炸或梯度消失。而 LSTM 则通过将前面时刻的记忆单元决策以及当前时刻的候选记忆单元决策合并在一起，从而降低了梯度消失的风险，较长的输入文本句子可以通过学习到的有效记忆单元状态进行处理，长期信息被记住，从而能够更好地处理输入文本。LSTM 通过引入门控机制，可以选择性地保留和遗忘信息，因此可以有效地缓解梯度消失的问题。因此在训练过程中，LSTM 的梯度能够更好地传递，使得网络在长序列上能够更好地收敛。

三、 自己实现的 LSTM

(一) 网络结构

```
class MyLSTM(nn.Module):
    def __init__(self, input_sz, hidden_sz):
        super().__init__()
        self.input_sz = input_sz
        self.hidden_size = hidden_sz
        self.W = nn.Parameter(torch.Tensor(input_sz, hidden_sz * 4))
        self.U = nn.Parameter(torch.Tensor(hidden_sz, hidden_sz * 4))
        self.bias = nn.Parameter(torch.Tensor(hidden_sz * 4))
        self.init_weights()

    def init_weights(self):
        stdv = 1.0 / math.sqrt(self.hidden_size)
        for weight in self.parameters():
            weight.data.uniform_(-stdv, stdv)
```

```
15
16      def forward(self, x,
17                  init_states=None):
18          """Assumes x is of shape (batch, sequence, feature)"""
19          bs, seq_sz, _ = x.size()
20          hidden_seq = []
21          if init_states is None:
22              h_t, c_t = (torch.zeros(bs, self.hidden_size).to(x.device),
23                          torch.zeros(bs, self.hidden_size).to(x.device))
24          else:
25              h_t, c_t = init_states
26
27          HS = self.hidden_size
28          for t in range(seq_sz):
29              x_t = x[:, t, :]
30              # batch the computations into a single matrix multiplication
31              gates = x_t @ self.W + h_t @ self.U + self.bias
32              i_t, f_t, g_t, o_t = (
33                  torch.sigmoid(gates[:, :HS]), # input
34                  torch.sigmoid(gates[:, HS:HS*2]), # forget
35                  torch.tanh(gates[:, HS*2:HS*3]),
36                  torch.sigmoid(gates[:, HS*3:]), # output
37              )
38              c_t = f_t * c_t + i_t * g_t
39              h_t = o_t * torch.tanh(c_t)
40              hidden_seq.append(h_t.unsqueeze(0))
41          hidden_seq = torch.cat(hidden_seq, dim=0)
42          # reshape from shape (sequence, batch, feature) to (batch, sequence,
                feature)
43          hidden_seq = hidden_seq.transpose(0, 1).contiguous()
44          return hidden_seq, (h_t, c_t)
```

(二) 实验结果

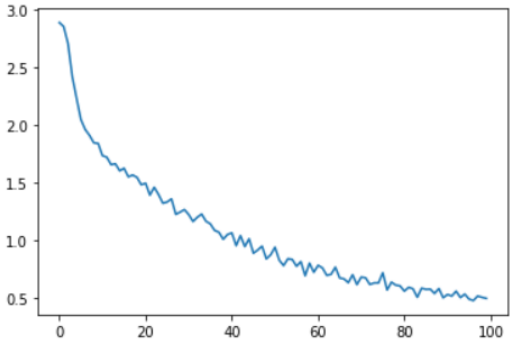

图 7: **自己实现 LSTM 的损失**

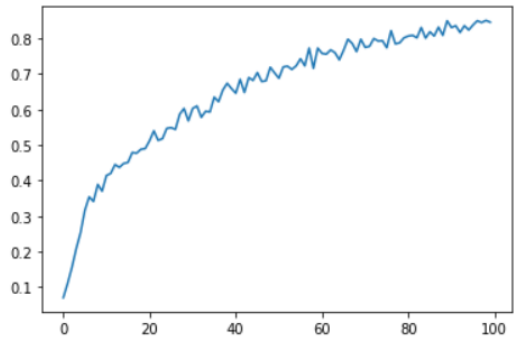

图 8: **自己实现 LSTM 的准确率**

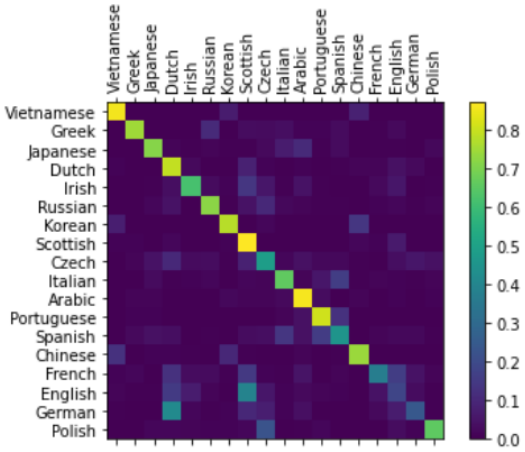

图 9: **自己实现 LSTM 的预测矩阵图**

可以看到效果接近原来 PYTORCH 自带的 LSTM。