RNN 掃盲：循環神經網絡解讀及其PyTorch 應用實現

新機器視覺 2022-03-28 21:00

來自| 知乎
作者 | Lucas
地址| https://zhuanlan.zhihu.com/p/85995376

RNN 掃盲：循環神經網絡解讀及其PyTorch 應用實現

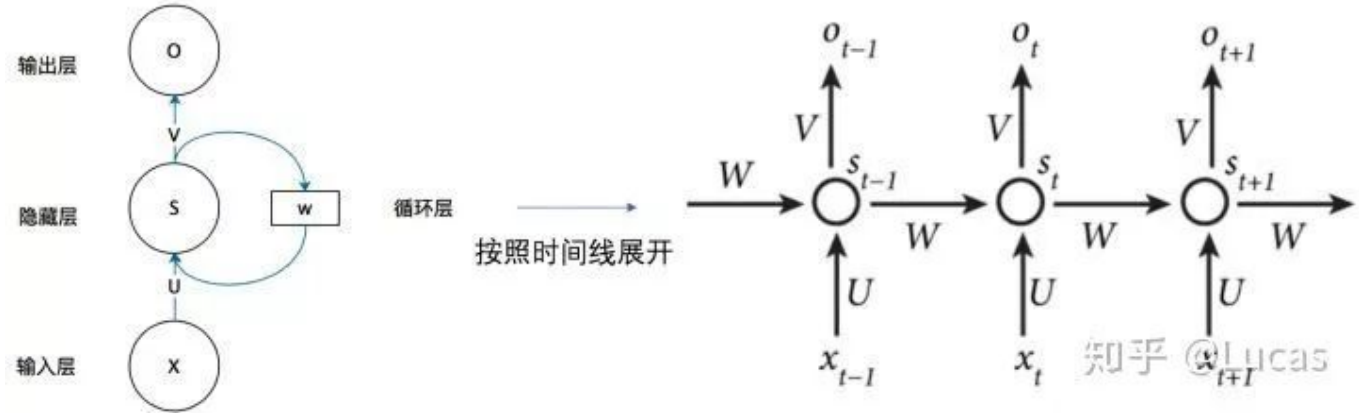
循環神經網絡(Recurrent Neural Network，RNN)是一類具有短期記憶能力的神經網絡。具體的表現形式為網絡會對前面的信息進行記憶並應用於當前輸出的計算中，也就是說隱藏層的輸入不僅包括輸入層的輸出還包括上一時刻隱藏層的輸出。簡單來說，設計RNN 就是為了處理序列數據。如果說CNN 是對人類視覺的仿真，那RNN 不妨先看作是對人類記憶能力的模擬。

為什麼需要RNN? 和CNN 的主要區別?

- CNN 相當於人類視覺，是沒有記憶能力的，沒有辦法根據以前的記憶來處理新任務。而RNN 是基於人的記憶的想法，期望網絡能夠機主前面出現的特徵，根據特徵完成下游任務。
- CNN 需要固定長度的輸入、輸出，RNN 的輸入和輸出可以是不定長且不等長的
- CNN 只有one-to-one 一種結構，而RNN 有多種結構。

結構組成

一個簡單RNN 由三個部分組成，輸入層、隱藏層，輸出層（廢話）如果我們把上面的圖展開，循環神經網絡也可以畫成下面這個樣子：



其中，t 是時刻， x 是輸入層， s 是隱藏層， o 是輸出層，矩陣W 就是隱藏層上一次的值作為這一次的輸入的權重。

為什麼循環神經網絡可以往前看任意多個輸入值呢?

看啊這個是輸出層o和隱藏層s 的計算公式

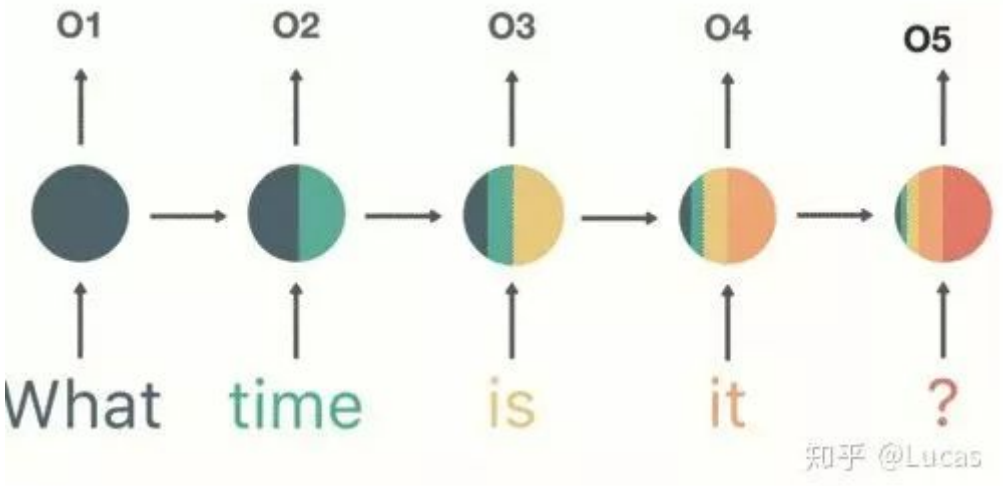
$$\mathbf{o}_t = g(V\mathbf{s}_t)$$
$$\mathbf{s}_t = f(U\mathbf{x}_t + W\mathbf{s}_{t-1})$$

如果把公式2 一直往公式1 裡帶，則有：

$$\begin{aligned}\mathbf{o}_t &= g(V\mathbf{s}_t)\\ &= Vf(U\mathbf{x}_t + W\mathbf{s}_{t-1})\\ &= Vf(U\mathbf{x}_t + Wf(U\mathbf{x}_{t-1} + W\mathbf{s}_{t-2}))\\ &= Vf(U\mathbf{x}_t + Wf(U\mathbf{x}_{t-1} + Wf(U\mathbf{x}_{t-2} + W\mathbf{s}_{t-3})))\\ &= Vf(U\mathbf{x}_t + Wf(U\mathbf{x}_{t-1} + Wf(U\mathbf{x}_{t-2} + Wf(U\mathbf{x}_{t-3} + \ldots))))\end{aligned}$$

記憶能力

該模型具有一定的記憶能力，能夠按時序依次處理任意長度的信息。前面的輸入對未來產生影響。什麼意思呢下圖所示。當我們將“ What time is it ?" 每個詞進入神經網絡後都會對下一個詞產生影響，

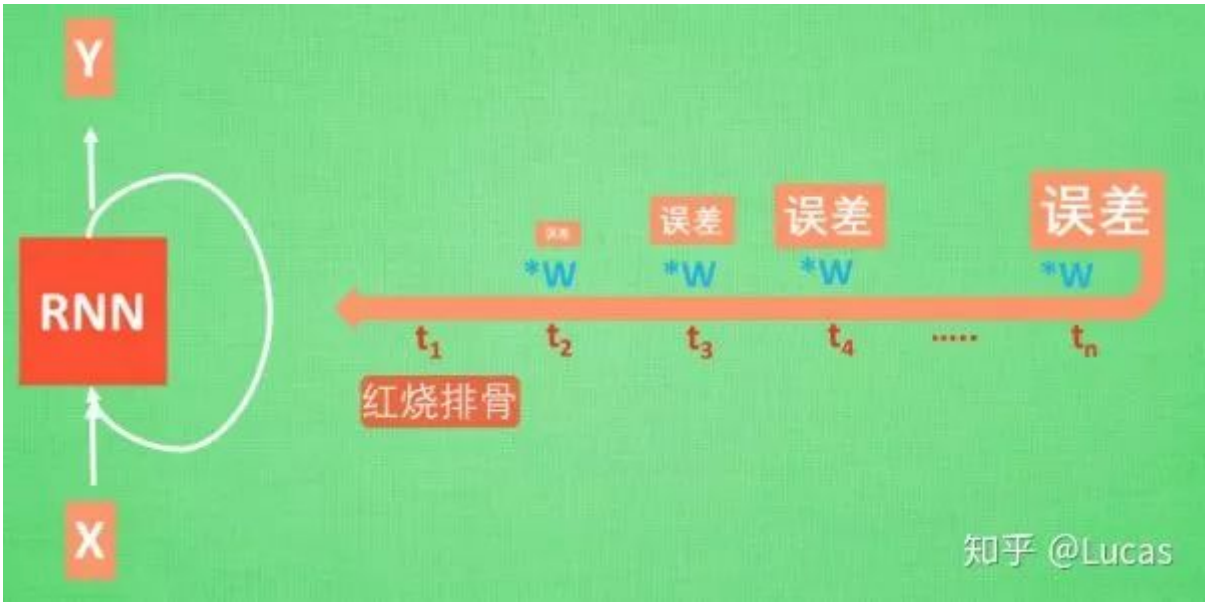


缺點：梯度消失和梯度爆炸

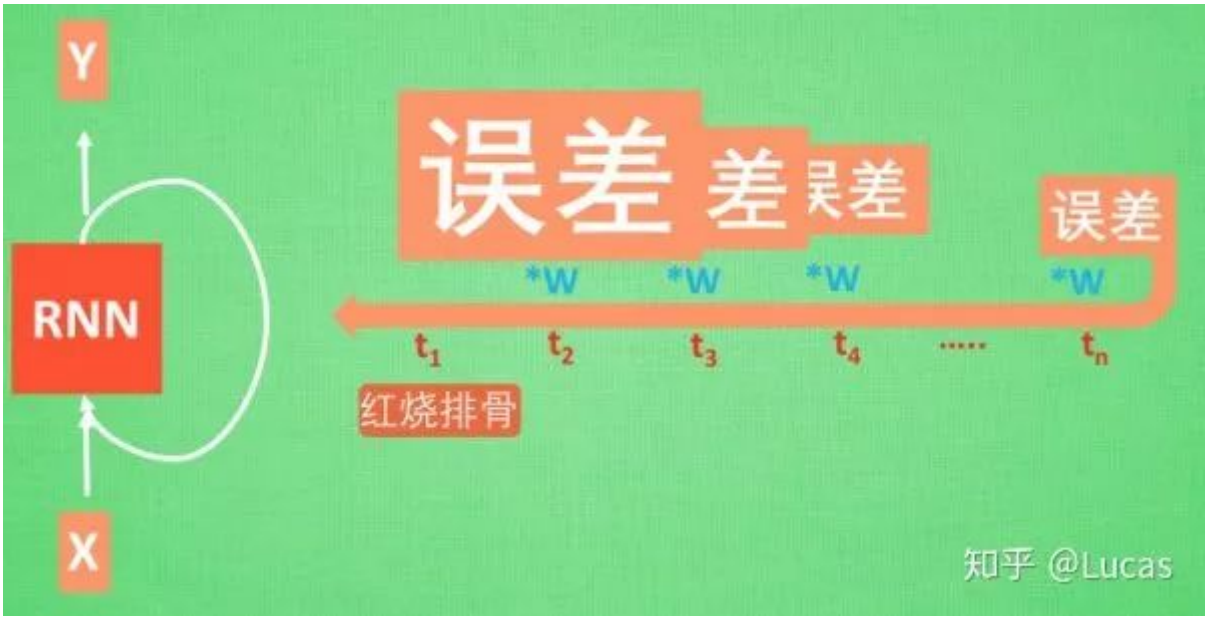
通過上面的例子，我們已經發現，短期的記憶影響較大（如橙色區域），但是長期的記憶影響就很小（如黑色和綠色區域），這就是RNN 存在的短期記憶問題。

莫煩Python 這裡講解的非常生動形象：

> '我今天要做紅燒排骨, 首先要準備排骨, 然後…., 最後美味的一道菜就出鍋了'。現在請RNN 來分析, 我今天做的到底是什麼菜呢. RNN可能會給出"辣子雞"這個答案. 由於判斷失誤, RNN就要開始學習這個長序列X 和'紅燒排骨' 的關係, 而RNN需要的關鍵信息"紅燒排骨"卻出現在句子開頭。



梯度消失示意



梯度爆炸示意

> 紅燒排骨這個信息原的記憶要進過長途跋涉才能抵達最後一個時間點. 然後我們得到誤差, 而且在反向傳遞得到的誤差的時候, 他在每一步都會乘以一個自己的參數W. 如果這個W 是一個小於1 的數, 比如0.9. 這個0.9 不斷乘以誤差, 誤差傳到初始時間點也會是一個接近於零的數, 所以對於初始時刻, 誤差相當於就消失了. 我們把這個問題叫做梯度消失或者梯度彌散Gradient vanishing. 反之如果W 是一個大於1 的數, 比如1.1 不斷累乘, 則到最後變成了無窮大的數, RNN被這無窮大的數撐死了, 這種情況我們叫做梯度爆炸, 這就是普通RNN 沒有辦法回憶起久遠記憶的原因.

在此感謝

@莫煩

前輩關於機器學習機器學習基礎知識的講解，讓晚輩能夠迅速成長，知識分享是人的一種高貴品質。我由衷感謝您!

基礎模型的PyTorch 實現

```
class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(RNN, self).__init__()

        self.input_size = input_size
        self.hidden_size = hidden_size
        self.output_size = output_size

        self.i2h = nn.Linear(input_size + hidden_size, hidden_size)
        self.i2o = nn.Linear(input_size + hidden_size, output_size)

    def forward(self, input, hidden):
        # 将input和之前的网络中的隐藏层参数合并。
        combined = torch.cat((input, hidden), 1)

        hidden = self.i2h(combined)  # 计算隐藏层参数
        output = self.i2o(combined)  # 计算网络输出的结果
        return output, hidden

    def init_hidden(self):
        # 初始化隐藏层参数hidden
        return torch.zeros(1, self.hidden_size)
```

參考資料

morvanzhou.github.io/tu

easyai.tech/ai-definiti

代碼地址：

github.com/zy1996code/n

—THE END—

喜歡此內容的人還喜歡

用機器學習神器sklearn做特徵工程！

新機器視覺

神經網絡訓練不收斂或訓練失敗的原因總結

計算機視覺life

理解卷積神經網絡中的自註意力機制

小白學視覺