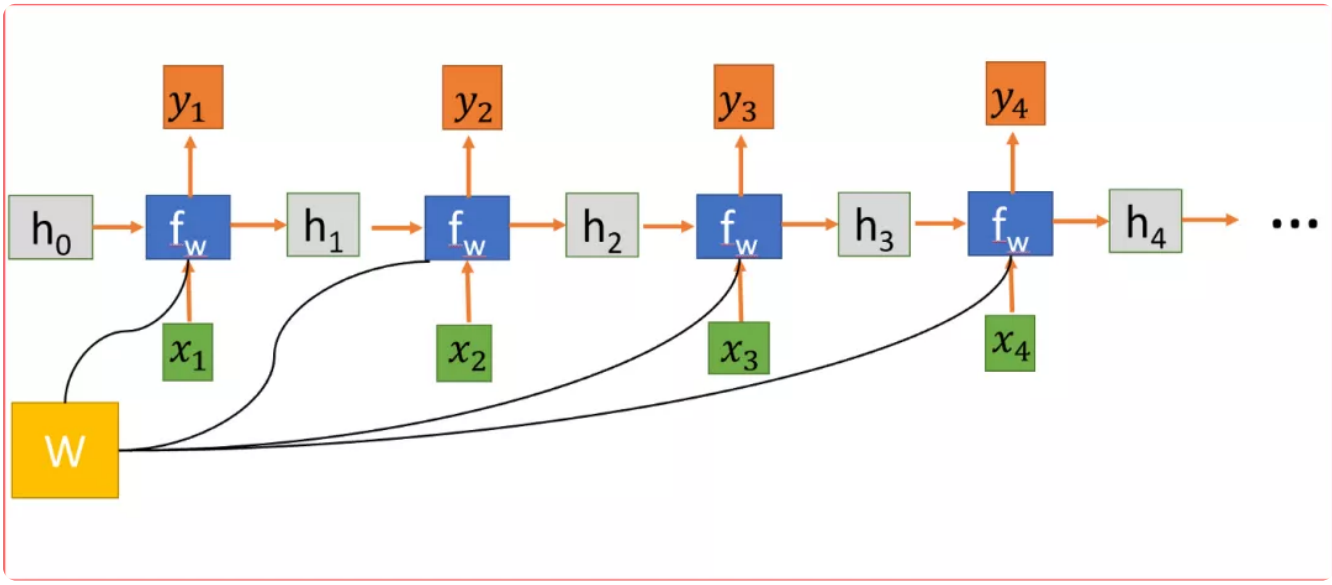
【深度学习】循环神经网络（RNN）简易教程

机器学习初学者 2020-07-26

以下文章来源于磐创AI ，作者VK

磐创AI
AI行业最新动态，机器学习干货文章，深度学习原创博客，深度学习实战项目，Tensorf...

文章来源于磐创AI，作者VK



磐创AI分享

作者 | Renu Khandelwal

编译 | VK

来源 | Medium

我们从以下问题开始

- 循环神经网络能解决人工神经网络和卷积神经网络存在的问题。
- 在哪里可以使用RNN？
- RNN是什么以及它是如何工作的？
- 挑战RNN的消梯度失和梯度爆炸
- LSTM和GRU如何解决这些挑战

假设我们正在写一条信息“Let’s meet for___”，我们需要预测下一个单词是什么。下一个词可以是午餐、晚餐、早餐或咖啡。我们更容易根据上下文作出推论。假设我们知道我们是在下午开会，并且这些信息一直存在于我们的记忆中，那么我们就可以很容易地预测我们可能会在午餐时见面。

当我们需要处理需要在多个时间步上的序列数据时，我们使用循环神经网络（RNN）

传统的神经网络和CNN需要一个固定的输入向量，在固定的层集上应用激活函数产生固定大小的输出。

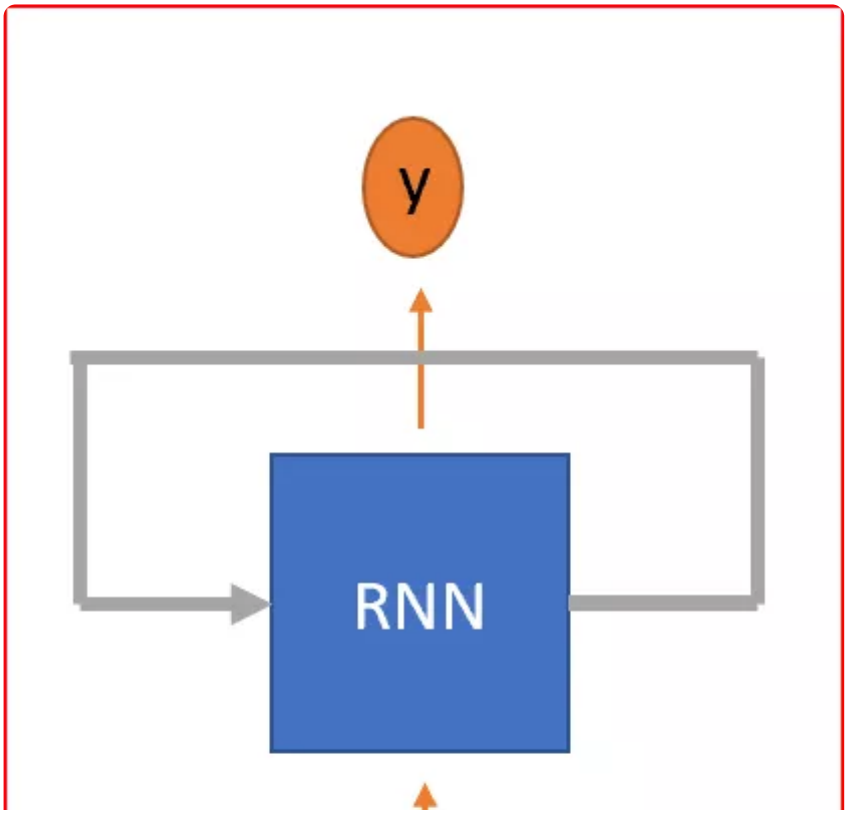
例如，我们使用128×128大小的向量的输入图像来预测狗、猫或汽车的图像。我们不能用可变大小的图像来做预测

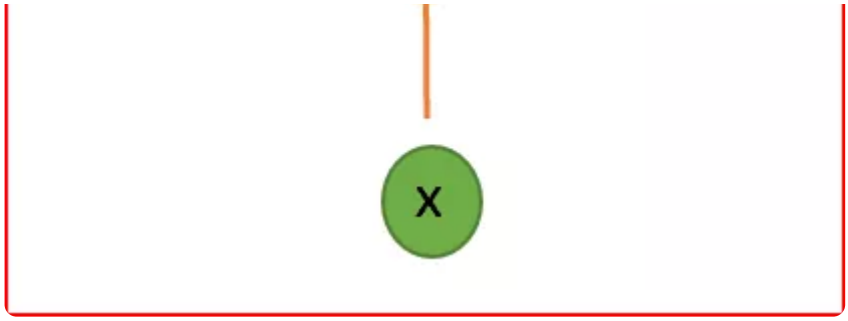
现在，如果我们需要对依赖于先前输入状态（如消息）的序列数据进行操作，或者序列数据可以在输入或输出中，或者同时在输入和输出中，而这正是我们使用RNNs的地方，该怎么办。

在RNN中，我们共享权重并将输出反馈给循环输入，这种循环公式有助于处理序列数据。

RNN利用连续的数据来推断谁在说话，说什么，下一个单词可能是什么等等。

RNN是一种神经网络，具有循环来保存信息。RNN被称为循环，因为它们对序列中的每个元素执行相同的任务，并且输出元素依赖于以前的元素或状态。这就是RNN如何持久化信息以使用上下文来推断。





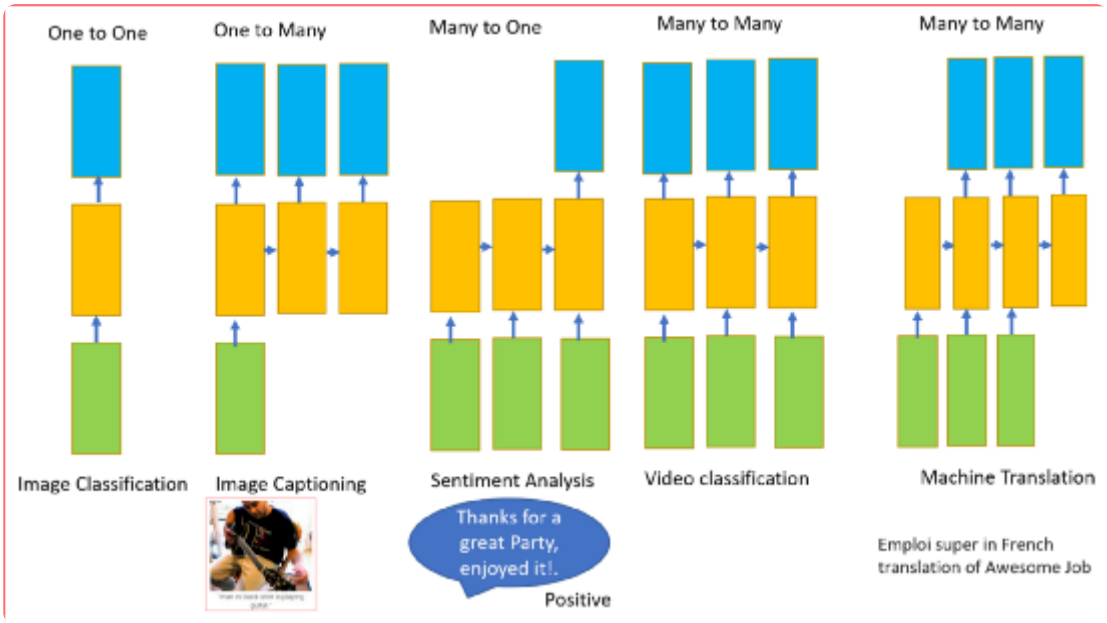
RNN是一种具有循环的神经网络

RNN在哪里使用？

前面所述的RNN可以有一个或多个输入和一个或多个输出，即可变输入和可变输出。

RNN可用于

- 分类图像
- 图像采集
- 机器翻译
- 视频分类
- 情绪分析

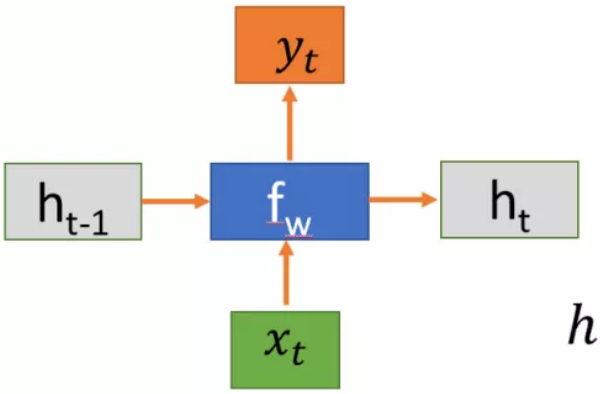


RNN是如何工作的？

先解释符号。

- h是隐藏状态
- x为输入
- y为输出
- W是权重
- t是时间步长

当我们在处理序列数据时，RNN在时间步t上取一个输入x。RNN在时间步t-1上取隐藏状态值来计算时间步t上的隐藏状态h并应用tanh激活函数。我们使用tanh或ReLU来表示输出和时间t的非线性关系。



$$h_t = f_\omega(h_{t-1}, x_t)$$

$$h_t = \tanh(w_{hh}h_{t-1} + w_{hx}x_t)$$

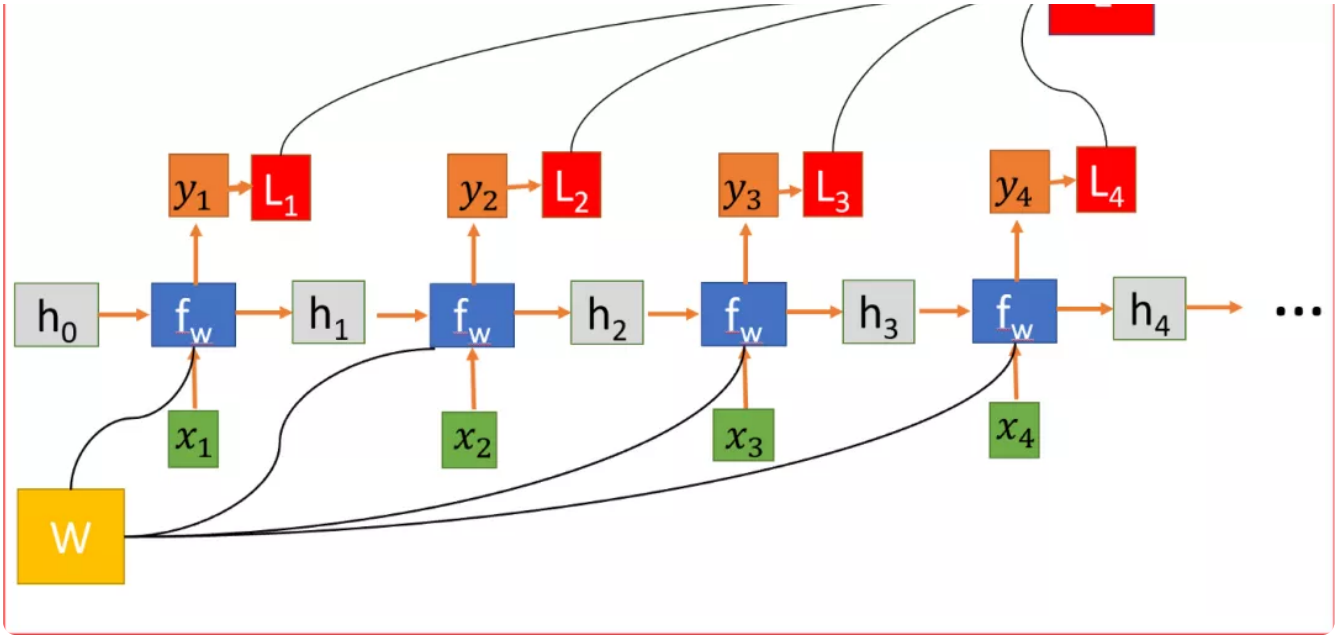
$$y_t = W_{hy}h_t$$

将RNN展开为四层神经网络，每一步共享权值矩阵W。

隐藏状态连接来自前一个状态的信息，因此充当RNN的记忆。任何时间步的输出都取决于当前输入以及以前的状态。

与其他对每个隐藏层使用不同参数的深层神经网络不同，RNN在每个步骤共享相同的权重参数。

我们随机初始化权重矩阵，在训练过程中，我们需要找到矩阵的值，使我们有理想的行为，所以我们计算损失函数L。损失函数L是通过测量实际输出和预测输出之间的差异来计算的。用交叉熵函数计算L。



RNN，其中损失函数L是各层所有损失的总和

为了减少损失，我们使用反向传播，但与传统的神经网络不同，RNN在多个层次上共享权重，换句话说，它在所有时间步骤上共享权重。这样，每一步的误差梯度也取决于前一步的损失。

在上面的例子中，为了计算第4步的梯度，我们需要将前3步的损失和第4步的损失相加。这称为通过Time-BPPT的反向传播。

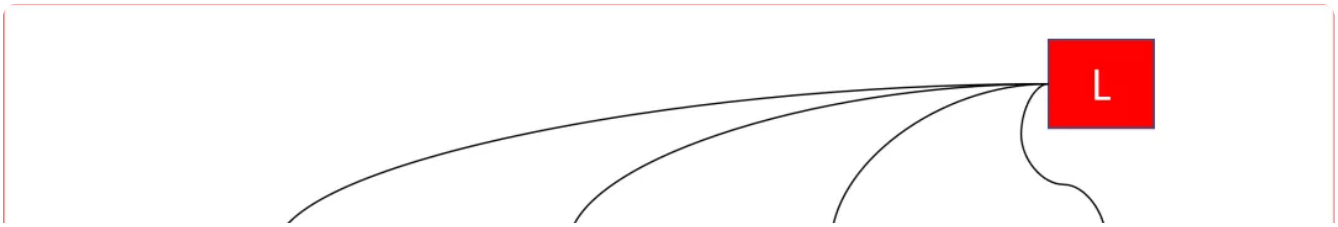
我们计算误差相对于权重的梯度，来为我们学习正确的权重，为我们获得理想的输出。

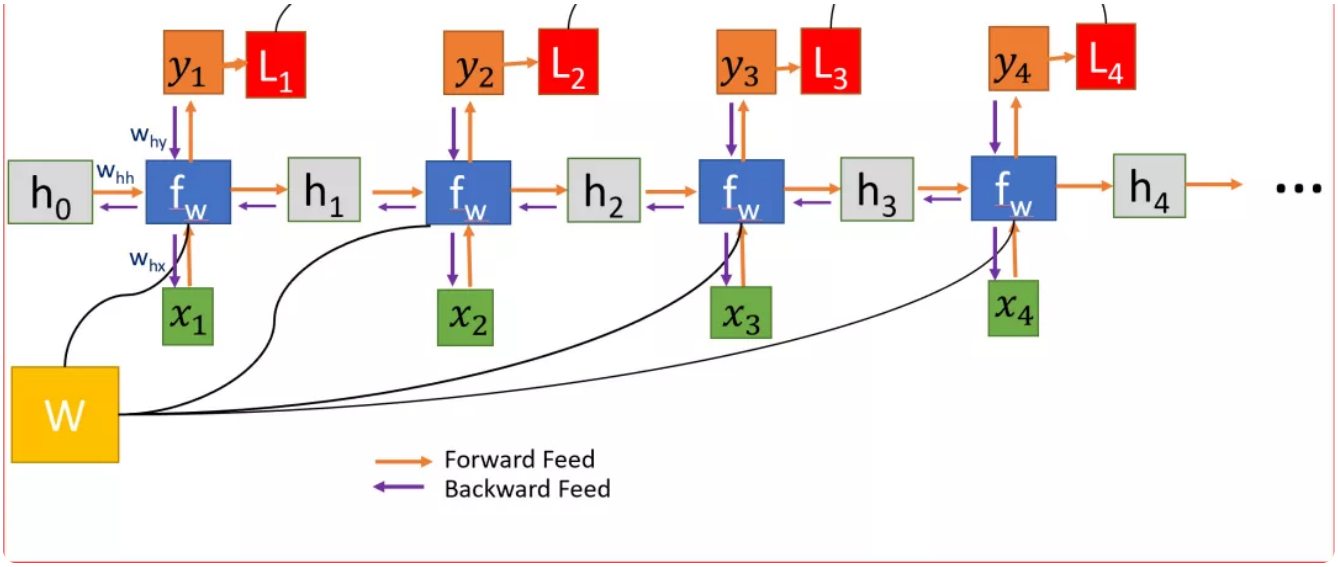
因为W在每一步中都被用到，直到最后的输出，我们从t=4反向传播到t=0。在传统的神经网络中，我们不共享权重，因此不需要对梯度进行求和，而在RNN中，我们共享权重，并且我们需要在每个时间步上对W的梯度进行求和。

在时间步t=0计算h的梯度涉及W的许多因素，因为我们需要通过每个RNN单元反向传播。即使我们不要权重矩阵，并且一次又一次地乘以相同的标量值，但是时间步如果特别大，比如说100个时间步，这将是一个挑战。

如果最大奇异值大于1，则梯度将爆炸，称为爆炸梯度。

如果最大奇异值小于1，则梯度将消失，称为消失梯度。



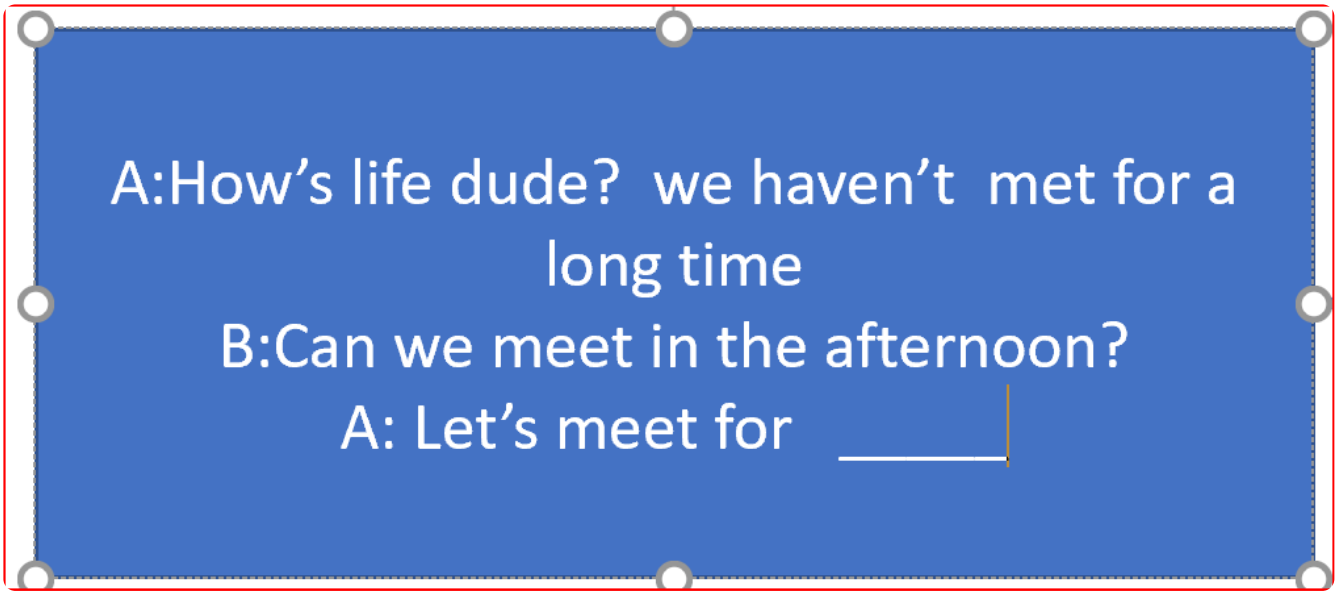


权重在所有层中共享，导致梯度爆炸或消失

对于梯度爆炸问题，我们可以使用梯度剪裁，其中我们可以预先设置一个阈值，如果梯度值大于阈值，我们可以剪裁它。

为了解决消失梯度问题，常用的方法是使用长短期记忆（LSTM）或门控循环单元（GRU）。

在我们的消息示例中，为了预测下一个单词，我们需要返回几个时间步骤来了解前面的单词。我们有可能在两个相关信息之间有足够的差距。随着差距的扩大，RNN很难学习和连接信息。但这反而是LSTM的强大功能。

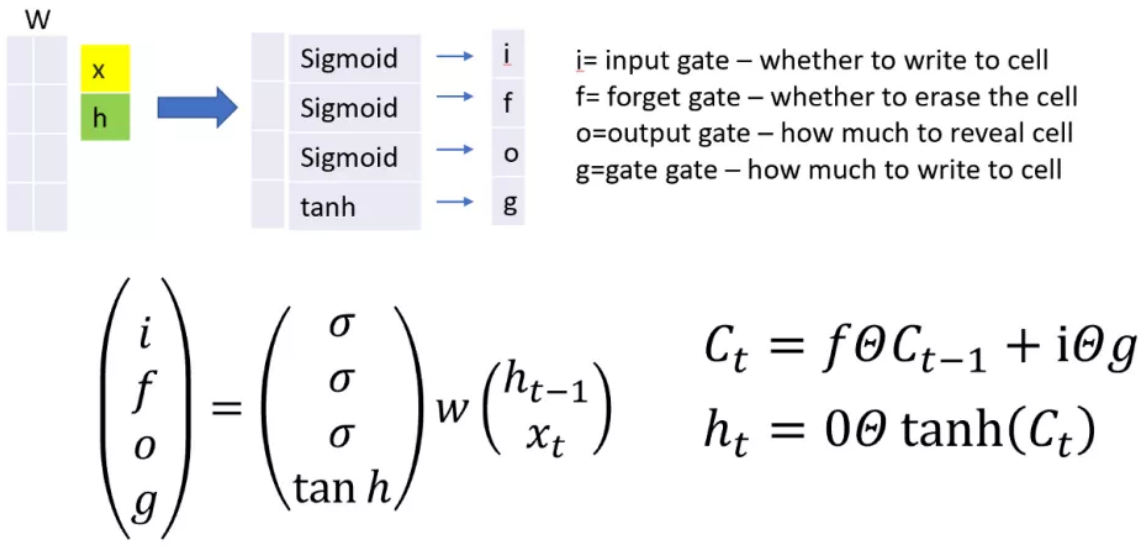


长短时记忆网络（LSTM）

LSTMs能够更快地学习长期依赖关系。LSTMs可以学习跨1000步的时间间隔。这是通过一种高效的基于梯度的算法实现的。

为了预测消息中的下一个单词，我们可以将上下文存储到消息的开头，这样我们就有了正确的上下文。这正是我们记忆的工作方式。

让我们深入了解一下LSTM架构，了解它是如何工作的

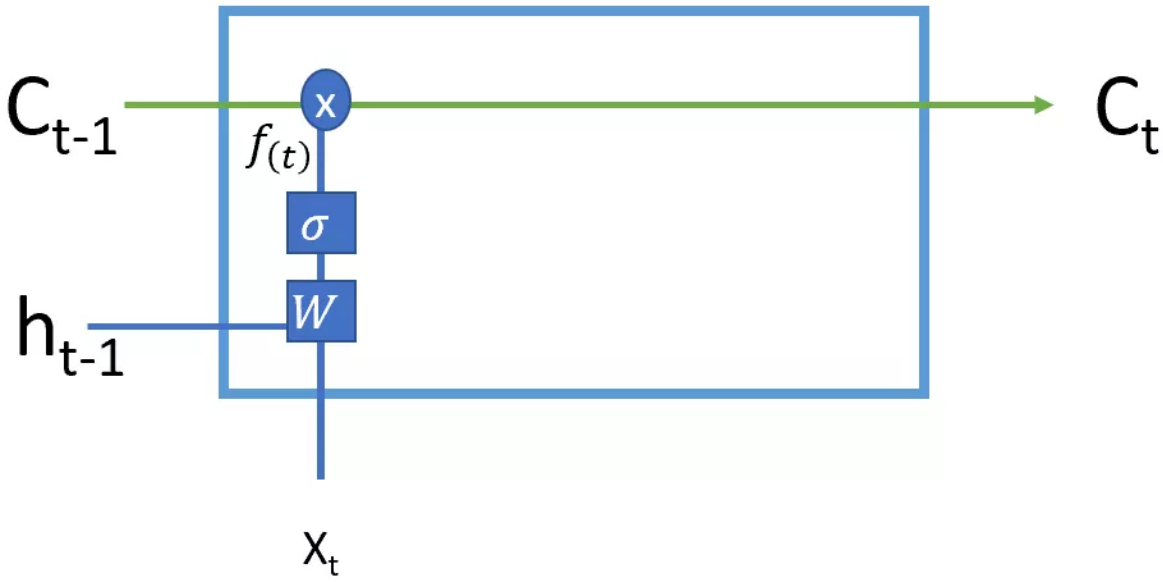


LSTMs的行为是在很长一段时间内记住信息，因此它需要知道要记住什么和忘记什么。

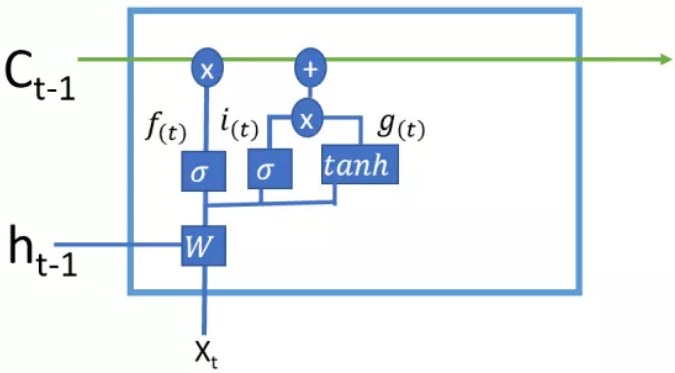
LSTM使用4个门，你可以将它们认为是否需要记住以前的状态。单元状态在LSTMs中起着关键作用。LSTM可以使用4个调节门来决定是否要从单元状态添加或删除信息。

这些门的作用就像水龙头，决定了应该通过多少信息。

$$f_{(t)} = \sigma\left(W_{fx}x_t + W_{fh}h_{t-1} + b_f\right)$$



1. LSTM的第一步是决定我们是需要记住还是忘记单元的状态。遗忘门使用Sigmoid激活函数，输出值为0或1。遗忘门的输出1告诉我们要保留该值，值0告诉我们要忘记该值。



$$C_t = f_t \Theta C_{t-1} + \mathrm{i}_t \Theta g_t$$

$$f_{(t)} = \sigma\left(W_{fx}x_t + W_{fh}h_{t-1} + b_f\right)$$
$$i_{(t)} = \sigma\left(W_{ix}x_t + W_{ix}h_{t-1} + b_i\right)$$
$$g_{(t)} = tanh\left(W_{gx}x_t + W_{gh}h_{t-1} + b_g\right)$$

2. 第二步决定我们将在单元状态中存储哪些新信息。这有两部分：一部分是输入门，它通过使用sigmoid函数决定是否写入单元状态；另一部分是使用tanh激活函数决定有哪些新信息被加入。

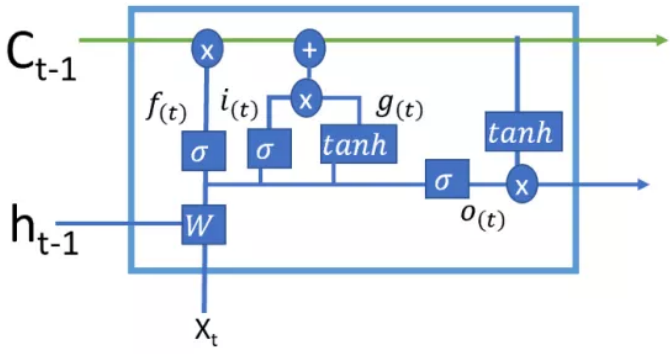
$$h_t = 0_t \Theta \tanh(C_t)$$

3. 在最后一步中，我们通过组合步骤1和步骤2的输出来创建单元状态，步骤1和步骤2的输出是将当前时间步的tanh激活函数应用于输出门的输出后乘以单元状态。Tanh激活函数给出-1和+1之间的输出范围

4. 单元状态是单元的内部存储器，它将先前的单元状态乘以遗忘门，然后将新计算的隐藏状态（g）乘以输入门i的输出。

$$C_t = f_t \Theta C_{t-1} + \mathrm{i}_t \Theta g_t$$

最后，输出将基于单元状态

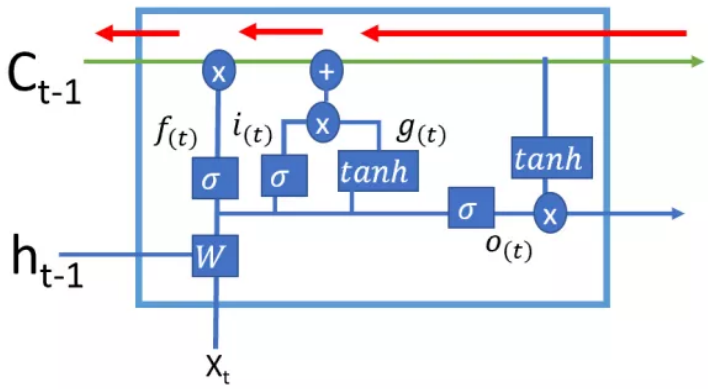


$$C_t = f_t \Theta C_{t-1} + \mathrm{i}_t \Theta g_t$$

$$h_t = 0_t \Theta \tanh(C_t)$$

$$f_{(t)} = \sigma(W_{fx} x_t + W_{fh} h_{t-1} + b_f)$$
$$i_{(t)} = \sigma(W_{ix} x_t + W_{ix} h_{t-1} + b_i)$$
$$g_{(t)} = tanh(W_{gx} x_t + W_{gh} h_{t-1} + b_g)$$
$$o_{(t)} = \sigma(W_{ox} x_t + W_{oh} h_{t-1} + b_o)$$

从当前单元状态到前一单元状态的反向传播只有遗忘门的单元相乘，没有W的矩阵相乘，这就利用单元状态消除了消失和爆炸梯度问题



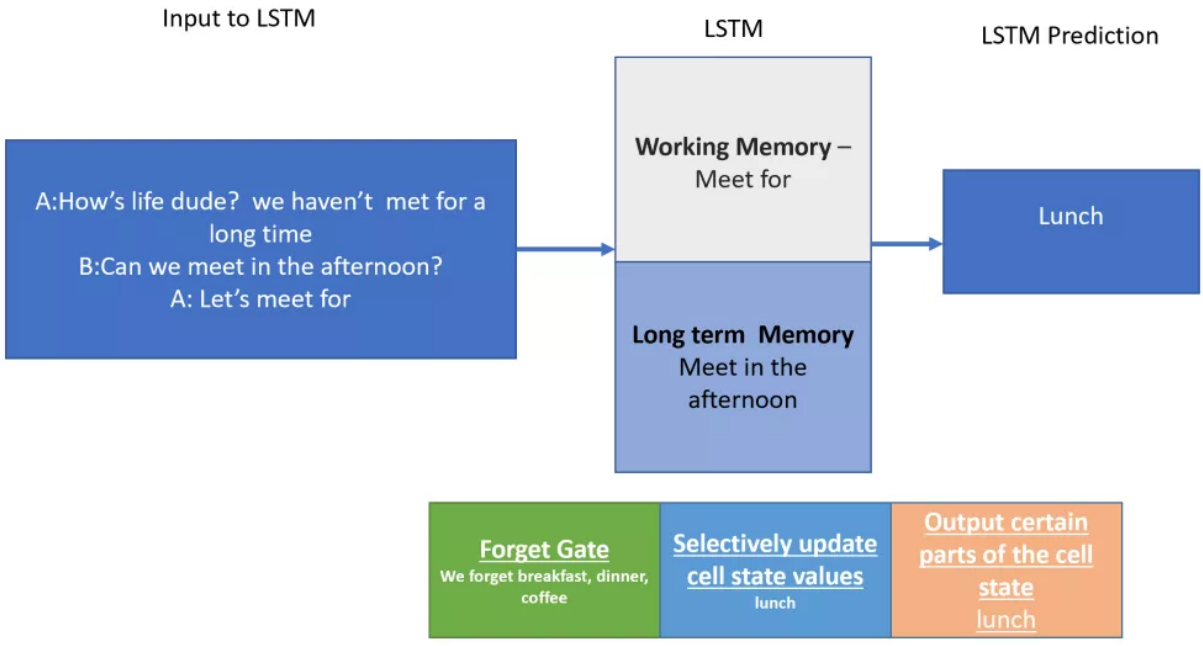
$$C_t = f_t \Theta C_{t-1} + i_t \Theta g_t$$

$$h_t = 0_t \Theta \tanh(C_t)$$

$$f_{(t)} = \sigma\left(W_{fx}x_t + W_{fh}h_{t-1} + b_f\right)$$
$$i_{(t)} = \sigma(W_{ix}x_t + W_{ix}h_{t-1} + b_i)$$
$$g_{(t)} = tanh\left(W_{gx}x_t + W_{gh}h_{t-1} + b_g\right)$$
$$o_{(t)} = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o)$$

LSTM通过决定忘记什么、记住什么、更新哪些信息来决定何时以及如何在每个时间步骤转换记忆。这就是LSTMs如何帮助存储长期记忆。

以下LSTM如何对我们的消息进行预测的示例



GRU，LSTM的变体

GRU使用两个门，重置门和一个更新门，这与LSTM中的三个步骤不同。GRU没有内部记忆

重置门决定如何将新输入与前一个时间步的记忆相结合。

更新门决定了应该保留多少以前的记忆。更新门是我们在LSTM中理解的输入门和遗忘门的组合。

GRU是求解消失梯度问题的LSTM的一个简单变种

原文链接：https://medium.com/datadriveninvestor/recurrent-neural-network-rnn-52dd4f01b7e8

✂----------

看到这里，说明你喜欢这篇文章，请点击「在看」或顺手「转发」「点赞」。

往期精彩回顾

- 适合初学者入门人工智能的路线及资料下载
- 机器学习及深度学习笔记等资料打印
- 机器学习在线手册
- 深度学习笔记专辑
- 《统计学习方法》的代码复现专辑