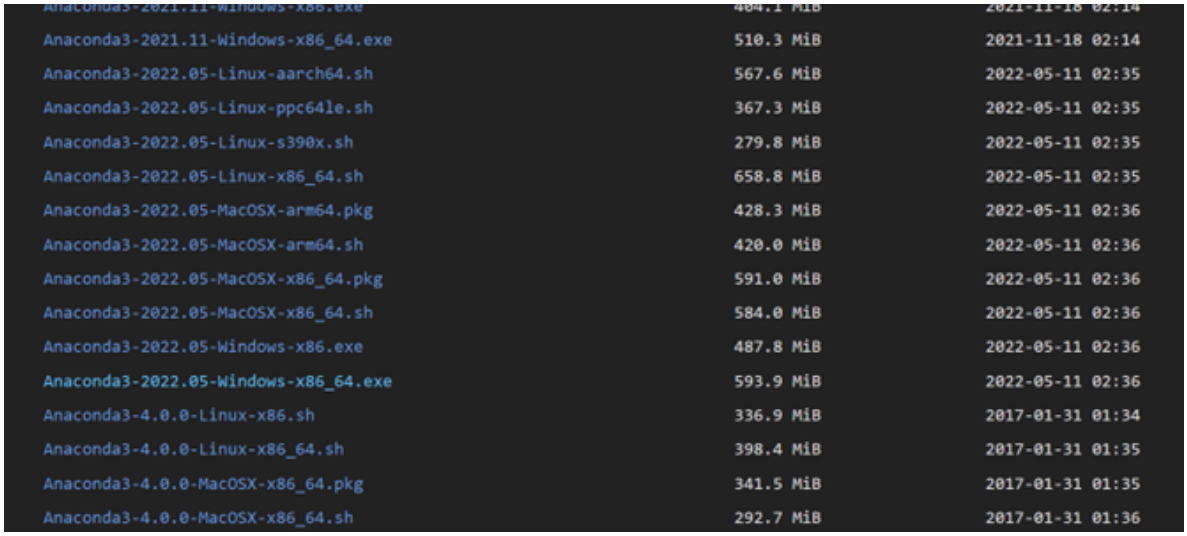
使用SentimentNet实现情感分类（基于华为云mindspore框架）

请同学们自行配置好自己的python工具，可以使用anaconda（推荐），jupyter或者pycharm。此处展示以anaconda作为内核的jupyter：

Annaconda安装：

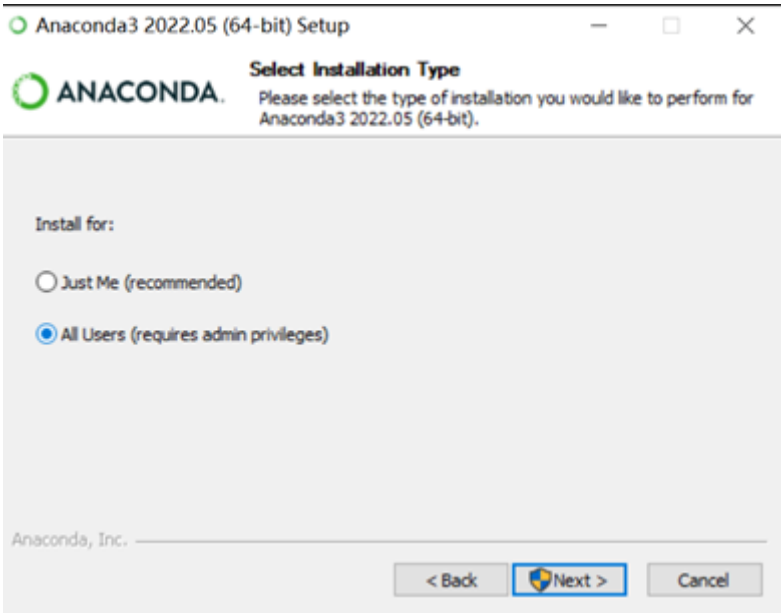
选择清华大学的独立镜像网站

根据自己的机器配置来选择下载版本:

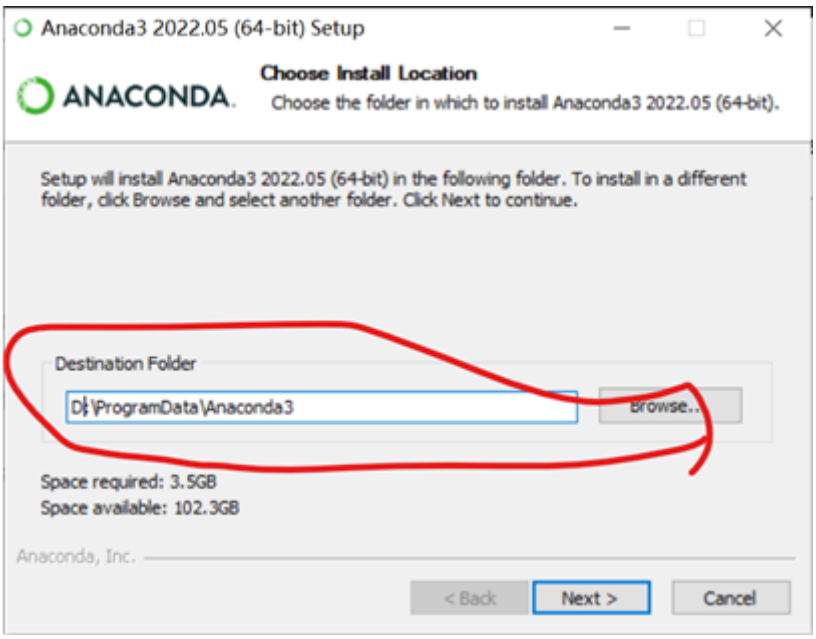


Anaconda3-2021.11-Windows-x86_64.exe	510.3 MiB	2021-11-18 02:14
Anaconda3-2022.05-Linux-aarch64.sh	567.6 MiB	2022-05-11 02:35
Anaconda3-2022.05-Linux-ppc64le.sh	367.3 MiB	2022-05-11 02:35
Anaconda3-2022.05-Linux-s390x.sh	279.8 MiB	2022-05-11 02:35
Anaconda3-2022.05-Linux-x86_64.sh	658.8 MiB	2022-05-11 02:35
Anaconda3-2022.05-MacOSX-arm64.pkg	428.3 MiB	2022-05-11 02:36
Anaconda3-2022.05-MacOSX-arm64.sh	420.0 MiB	2022-05-11 02:36
Anaconda3-2022.05-MacOSX-x86_64.pkg	591.0 MiB	2022-05-11 02:36
Anaconda3-2022.05-MacOSX-x86_64.sh	584.0 MiB	2022-05-11 02:36
Anaconda3-2022.05-Windows-x86.exe	487.8 MiB	2022-05-11 02:36
Anaconda3-2022.05-Windows-x86_64.exe	593.9 MiB	2022-05-11 02:36
Anaconda3-4.0.0-Linux-x86.sh	336.9 MiB	2017-01-31 01:34
Anaconda3-4.0.0-Linux-x86_64.sh	398.4 MiB	2017-01-31 01:35
Anaconda3-4.0.0-MacOSX-x86_64.pkg	341.5 MiB	2017-01-31 01:35
Anaconda3-4.0.0-MacOSX-x86_64.sh	292.7 MiB	2017-01-31 01:36

点击next安装，选择all user:



选择安装路径（c盘有地方可以安装到c盘中）：



点击next直至安装完成。

添加系统环境变量：

> $path of Anaconda$ \Scripts
>
> $path of Anaconda$ \mingw-w64\bin
>
> $path of Anaconda$ \Library\usr\bin
>
> $path of Anaconda$ \Library\bin

打开cmd，输入conda –version检查conda环境是否配好（显示版本号即成功）。

Mindspore框架配置：

创建conda虚拟环境

```
conda cerate -n $your env name$ python=3.7.5
conda activate $your env name$
```

进入华为mindspore官网，选择1.2.0版本的mindspore并下载镜像：

			linux_x86_64.whl	f979dd0444769e4c6a77fc11880d
	CPU	Ubuntu-x86	mindspore-1.2.0-cp37-cp37m-linux_x86_64.whl	92421a45b0e5352621b6d17bcd6deafdbc9965b7ecd9f1219b83a8c02384c8d3
		Ubuntu-aarch64	mindspore-1.2.0-cp37-cp37m-linux_aarch64.whl	8042752a39c92fe39efc2208e236a3f989a4bb3d0ab4543b364d00fa79f11913
		Windows-x64	mindspore-1.2.0-cp37-cp37m-win_amd64.whl	6038b1c28d574c565bf6a62a317421418960ee7df03bca9487d8f7c909ddb208

进入.whl所在的文件目录，执行

```
pip install mindspore-1.2.0-cp37-cp37m-win_amd64.whl
```

检查是否安装成功：

```
python -c "import mindspore;mindspore.run_check()"
```

如果输出：

```
MindSpore version: 版本号
The result of multiplication calculation is correct, MindSpore has been
installed successfully!
```

则表示安装成功。

jupyter配置环境kernels:

```
conda deactivate
```

```
conda install ipykernel
```

```
conda activate $your env name$
```

```
conda install ipykernel
```

```
conda install nb_conda
```

```
python -m ipykernel install --user --name nlp(你的环境名) --display-name nlp（显示名字）
```

查看可用的kernel内核:

```
jupyter kernelspec list
```

打开jupyter lab

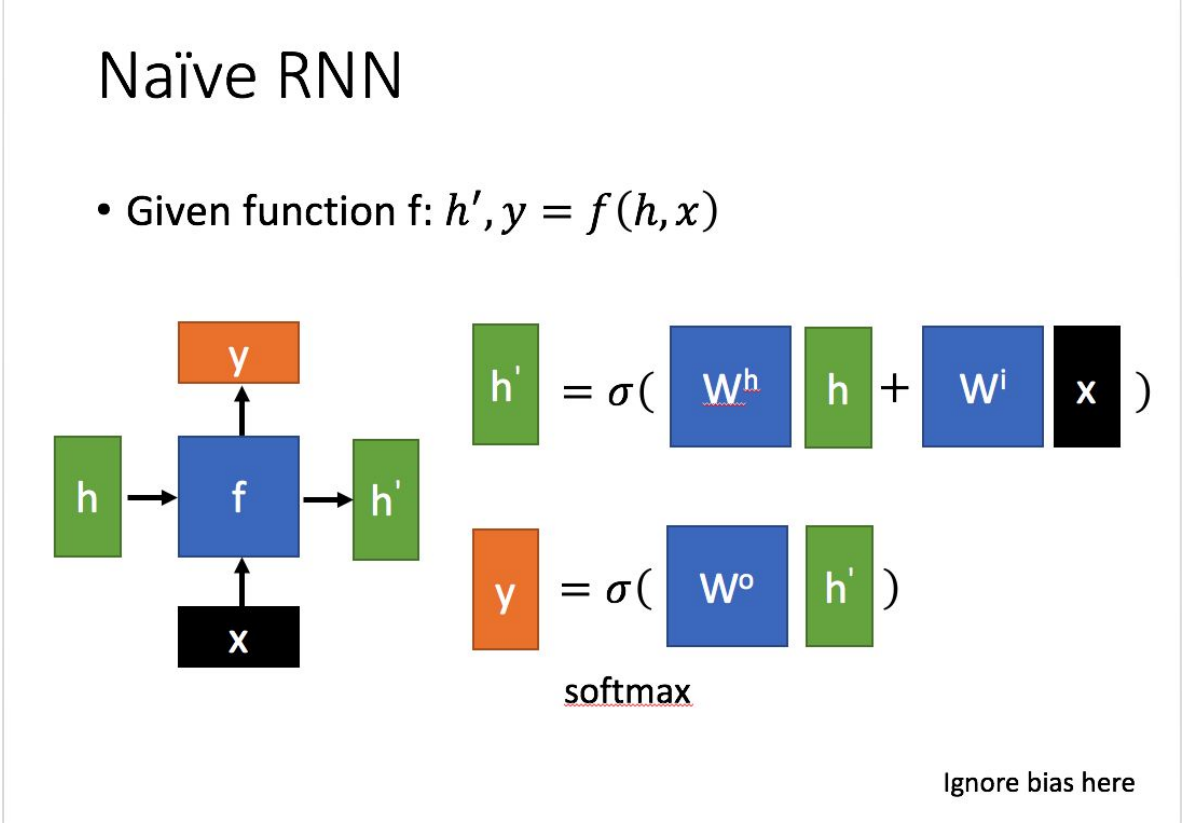
```
jupyter lab
```

LSTM简介：

1. 普通RNN

先简单介绍一下一般的RNN。

其主要形式如下图所示（图片均来自台大李宏毅教授的PPT）：



这里：

x 为当前状态下数据的输入， h 表示接收到的上一个节点的输入。

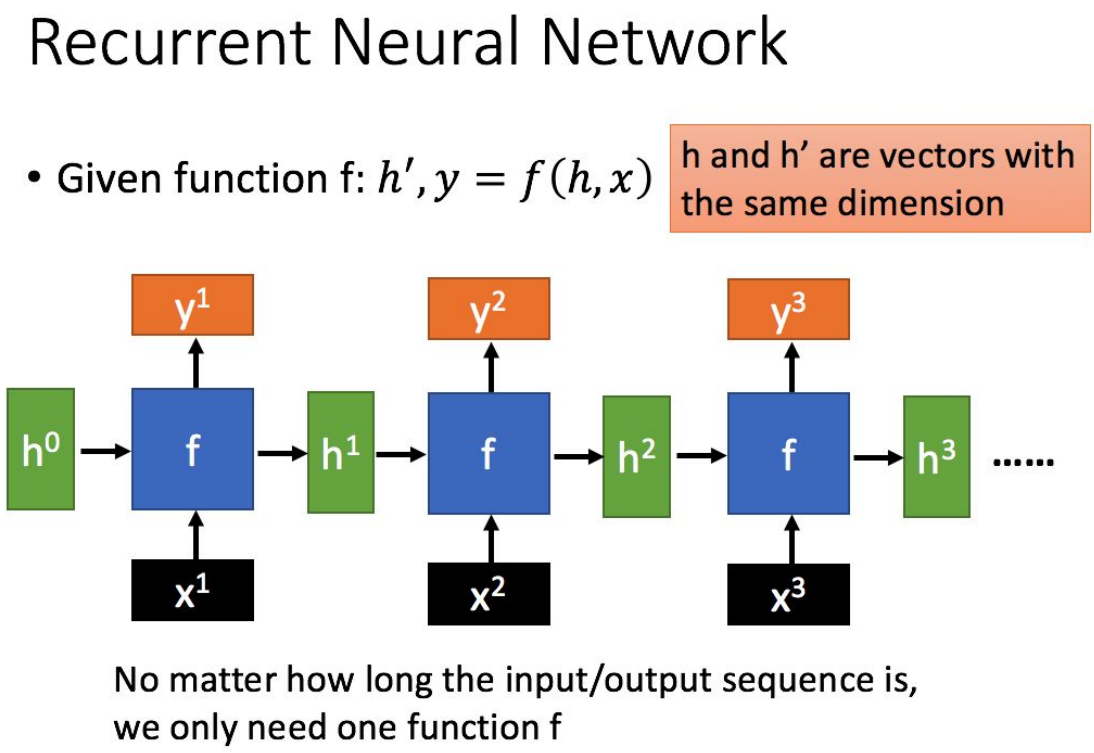
y 为当前节点状态下的输出，而 h' 为传递到下一个节点的输出。

通过上图的公式可以看到，输出 **h'** 与 **x** 和 **h** 的值都相关。

而 **y** 则常常使用 **h'** 投入到一个线性层（主要是进行维度映射）然后使用softmax进行分类得到需要的数据。

对这里的**y**如何通过 **h'** 计算得到往往看具体模型的使用方式。

通过序列形式的输入，我们能够得到如下形式的RNN。

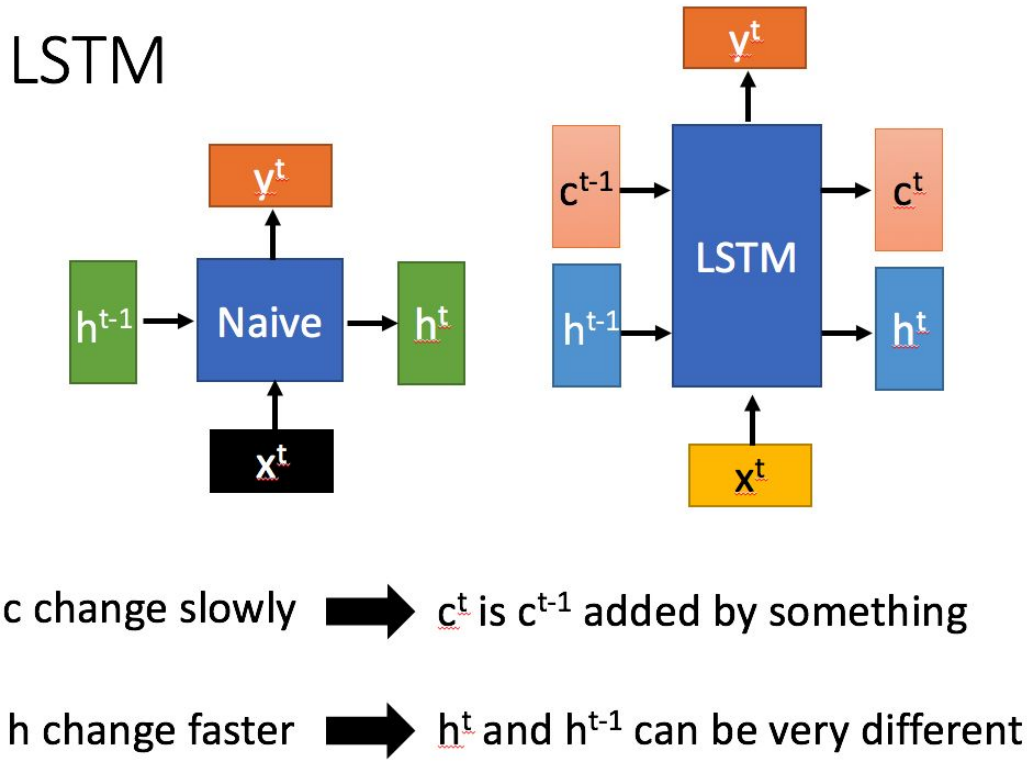


2. LSTM

2.1 什么是LSTM

长短期记忆（Long short-term memory, LSTM）是一种特殊的RNN，主要是为了解决长序列训练过程中的梯度消失和梯度爆炸问题。简单来说，就是相比普通的RNN，LSTM能够在更长的序列中有更好的表现。

LSTM结构（图右）和普通RNN的主要输入输出区别如下所示。



相比RNN只有一个传递状态 h^t ，LSTM有两个传输状态，一个 c^t （cell state），和一个 h^t （hidden state）。（Tips：RNN中的 h^t 对于LSTM中的 c^t ）

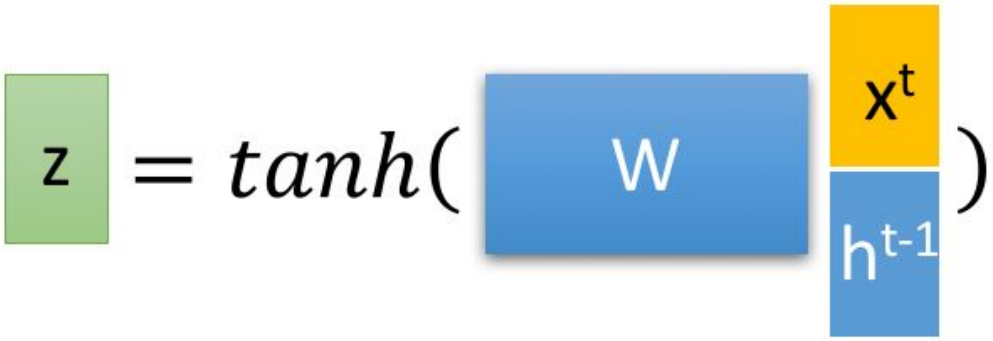
其中对于传递下去的 c^t 改变得很慢，通常输出的 c^t 是上一个状态传过来的 c^{t-1} 加上一些数值。

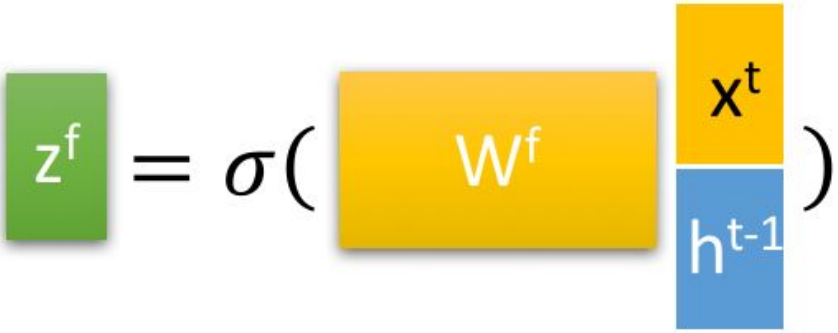
而 h^t 则在不同节点下往往会有很大的区别。

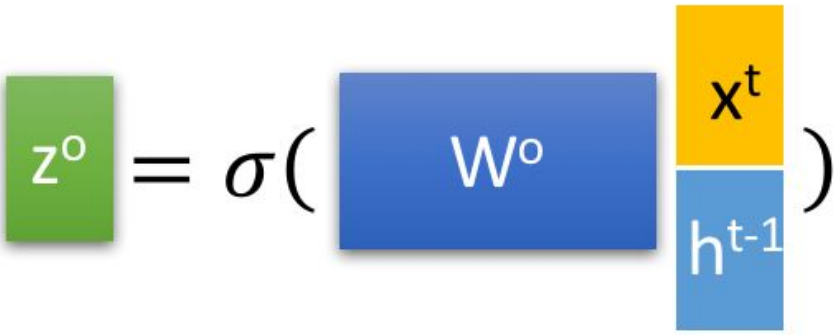
2.2 深入LSTM结构

下面具体对LSTM的内部结构来进行剖析。

首先使用LSTM的当前输入 x^t 和上一个状态传递下来的 h^{t-1} 拼接训练得到四个状态。

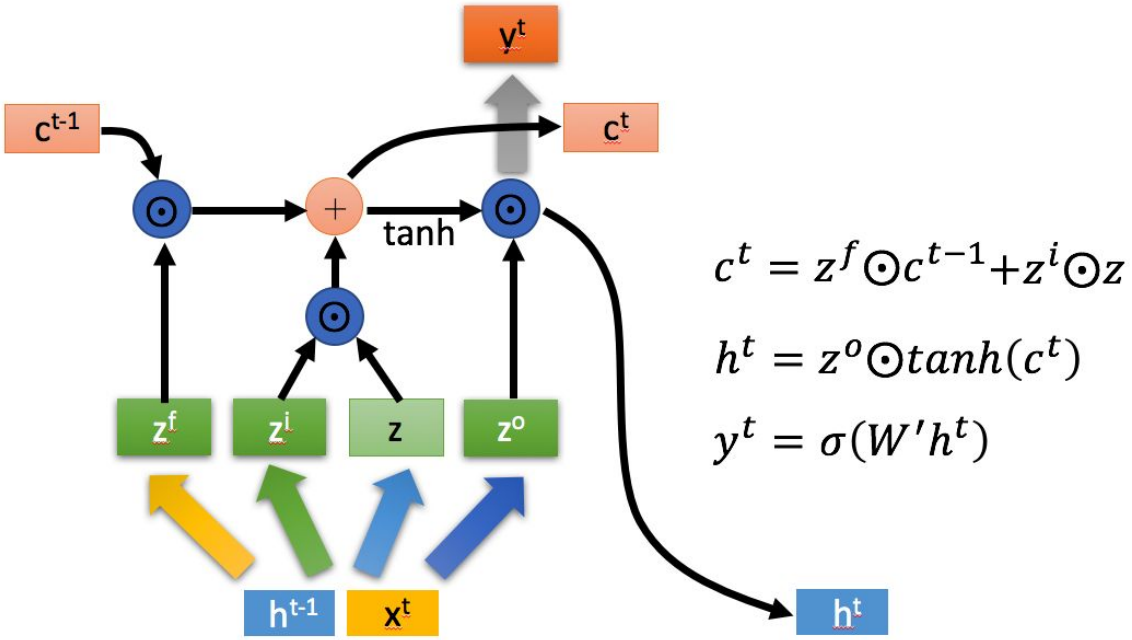






其中，z^f ， z^i ， z^o 是由拼接向量乘以权重矩阵之后，再通过一个 $sigmoid$ 激活函数转换成0到1之间的数值，来作为一种门控状态。而 z 则是将结果通过一个 $tanh$ 激活函数将转换成-1到1之间的值（这里使用 $tanh$ 是因为这里是将其做为输入数据，而不是门控信号）。

下面介绍这四个状态在LSTM内部的使用



$$c^t = z^f \odot c^{t-1} + z^i \odot z$$

$$h^t = z^o \odot tanh(c^t)$$

$$y^t = \sigma(W' h^t)$$

$\odot$ 是Hadamard Product，也就是操作矩阵中对应的元素相乘，因此要求两个相乘矩阵是同型的。 $\oplus$ 则代表进行矩阵加法。

LSTM内部主要有三个阶段：

1. 忘记阶段。这个阶段主要是对上一个节点传进来的输入进行**选择性**忘记。简单来说就是会 “忘记不重要的，记住重要的”。

具体来说是通过计算得到的 z^f （f表示forget）来作为忘记门控，来控制上一个状态的 c^{t-1} 哪些需要留哪些需要忘。

2. 选择记忆阶段。这个阶段将这个阶段的输入有选择性地进行“记忆”。主要是会对输入 x^t 进行选择记忆。哪些重要则着重记录下来，哪些不重要，则少记一些。当前的输入内容由前面计算得到的 z 表示。而选择的门控信号则是由 z^i （i代表information）来进行控制。

> 将上面两步得到的结果相加，即可得到传输给下一个状态的 c^t 。也就是上图中的第一个公式。

3. 输出阶段。这个阶段将决定哪些将会被当成当前状态的输出。主要是通过 z^o 来进行控制的。并且还对上一阶段得到的 c^o 进行了放缩（通过一个tanh激活函数进行变化）。

与普通RNN类似，输出 y^t 往往最终也是通过 h^t 变化得到。