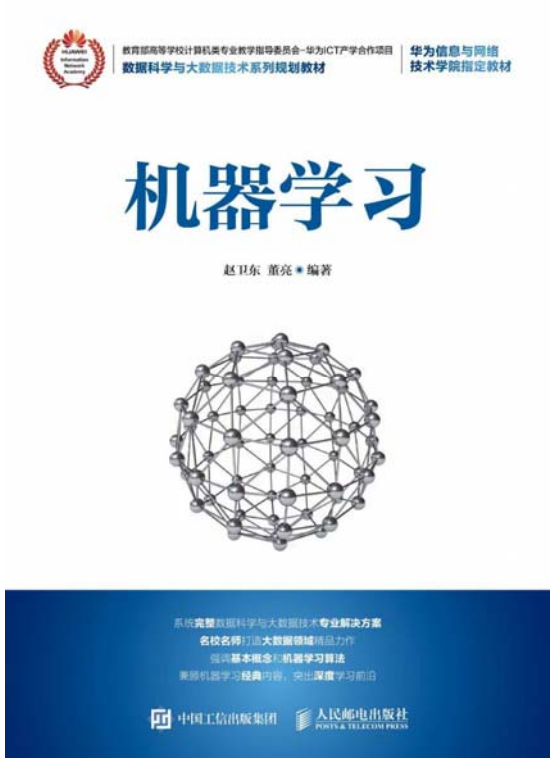


机器学习
循环神经网络

复旦大学 **赵卫东** 博士

wdzhao@fudan.edu.cn

章节介绍

- 深度学习是一种利用复杂结构的多个处理层来实现对数据进行高层次抽象的算法，是机器学习的一个重要分支。传统的BP算法仅有几层网络，需要手工指定特征且易出现局部最优问题，而深度学习引入了概率生成模型，可自动地从训练集提取特征，解决了手工特征考虑不周的问题，而且初始化了神经网络权重，采用反向传播算法进行训练，与BP算法相比取得了很好的效果。本章主要介绍了深度学习相关的概念和主流框架，重点介绍卷积神经网络和循环神经网络的结构以及常见应用。

章节结构

- 卷积神经网络
  - 卷积神经网络的结构
  - 常见卷积神经网络
- 循环神经网络
  - RNN基本原理
  - 长短期记忆网络
  - 门限循环单元

循环神经网络

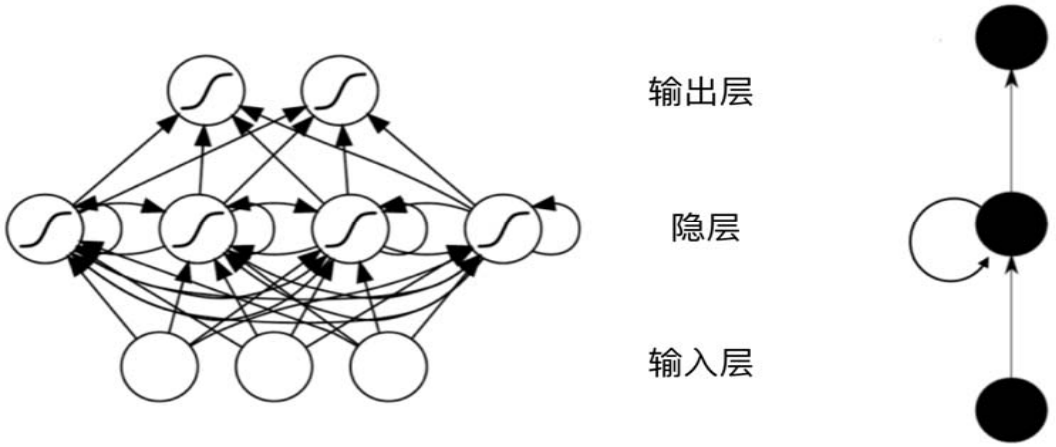
- 循环神经网络是一种对序列数据建模的神经网络。RNN不同于前向神经网络，它的层内、层与层之间的信息可以双向传递，更高效地存储信息，利用更复杂的方法来更新规则，通常用于处理信息序列的任务。RNN在自然语言处理、图像识别、语音识别、上下文的预测、在线交易预测、实时翻译等领域得到了大量的应用。

RNN基本原理

- RNN主要用来处理序列数据，在传统的神经网络模型中，是从输入层到隐含层再到输出层，每层内的节点之间无连接，循环神经网络中一个当前神经元的输出与前面的输出也有关，网络会对前面的信息进行记忆并应用于当前神经元的计算中，隐藏层之间的节点是有连接的，并且隐藏层的输入不仅包含输入层的输出还包含上一时刻隐藏层的输出。理论上，RNN可以对任意长度的序列数据进行处理。

RNN基本原理

- 一个典型的RNN网络结构如下图所示。



RNN基本原理

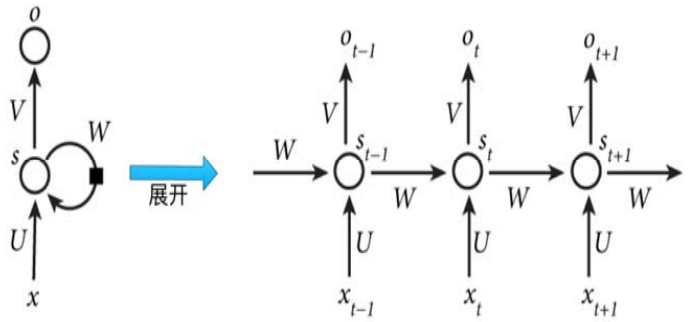
- RNN包含输入单元，输入集标记为x_t，而输出单元的输出集则被标记为y_t。RNN还包含隐藏单元，这些隐藏单元完成了主要工作。在某些情况下，RNN会引导信息从输出单元返回隐藏单元，并且隐藏层内的节点可以自连也可以互连。RNN的基本结构可以用以下公式表示。

$$h_t = f_w(h_{t-1}, x_t)$$

- 其中h_t表示新的目标状态，而h_{t-1}则是前一状态，x_t是当前输入向量，f_w是权重参数函数，目标值的结果与当前的输入、上一状态的结果有关系，以此可以求出各参数的权重值。

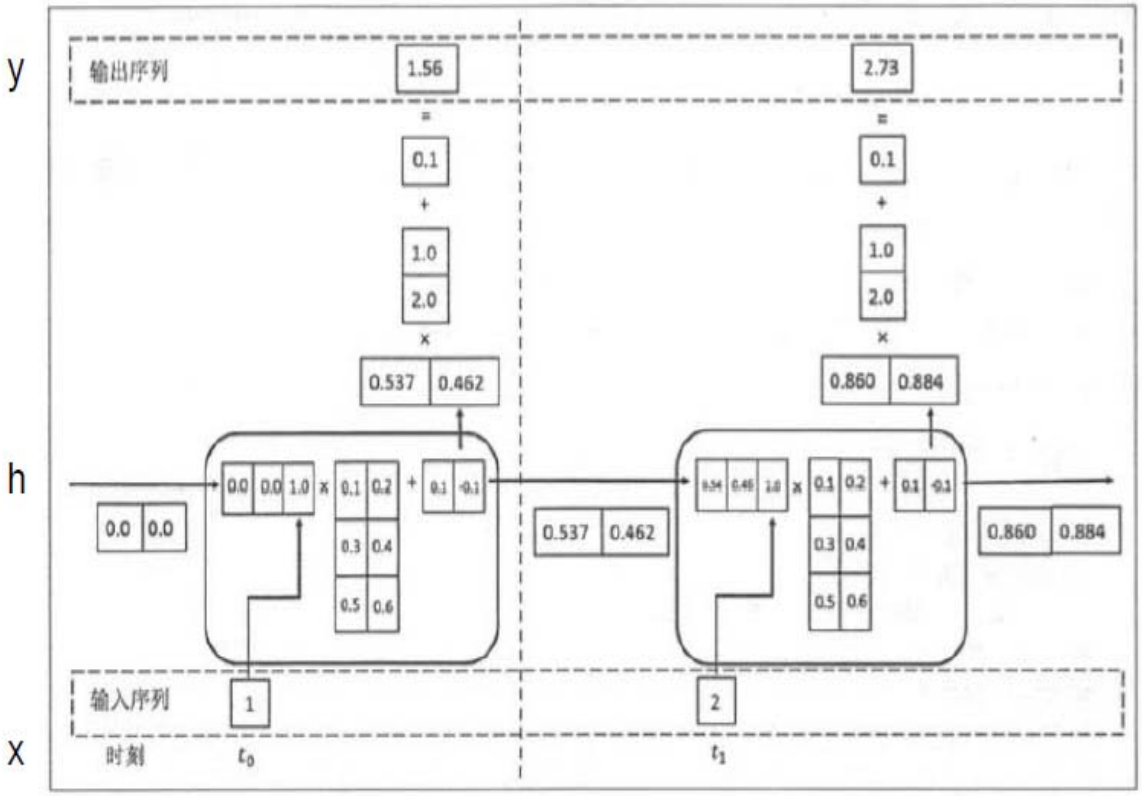
RNN基本原理

- 一个RNN可认为是同一网络的多次重复执行，每一次执行的结果是下一次执行的输入。循环神经网络展开图如右图所示。其中x_t是输入序列，s_t是在t时间步时隐藏状态，可以认为是网络的记忆，计算公式为$s_t = f(Ux_t + W_{s_{t-1}})$，其中$f$为非线性激活函数（如ReLU），$U$为当前输入的权重矩阵，$W$为上一状态的输入的权重矩阵，可以看到当前状态$s_t$依赖于上一状态$s_{t-1}$。



- 与CNN一样，RNN也是参数共享，在时间维度上，共享权重参数U、V和W

RNN示例



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

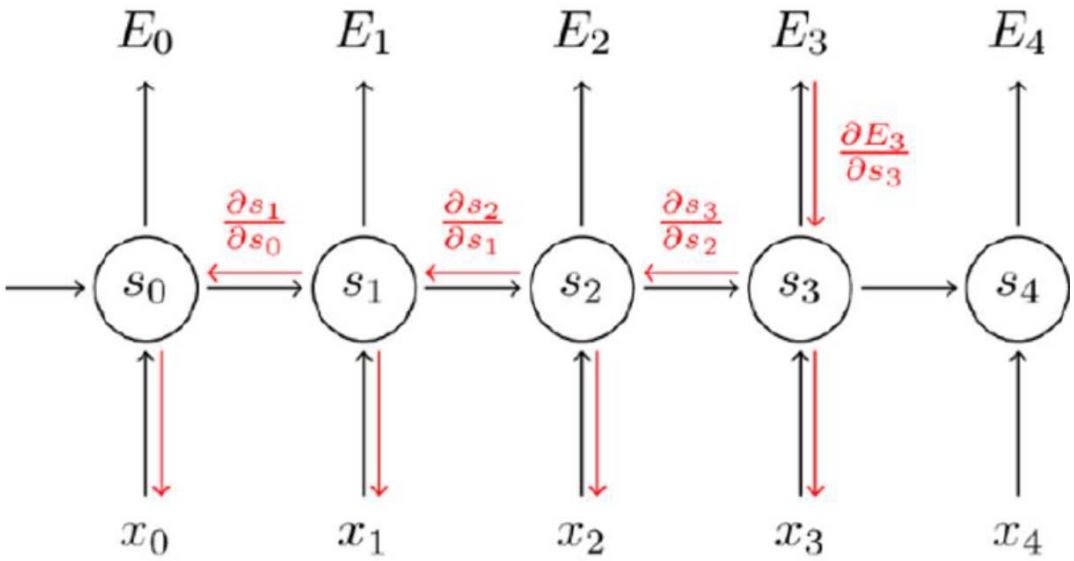
$$= \tanh([h_{t-1}, x_t]W + b_h)$$

$$y_t = softmax(w_{hy}h_t + b_y)$$

tanh([0.6, 0.5]) = [0.537, 0.462]

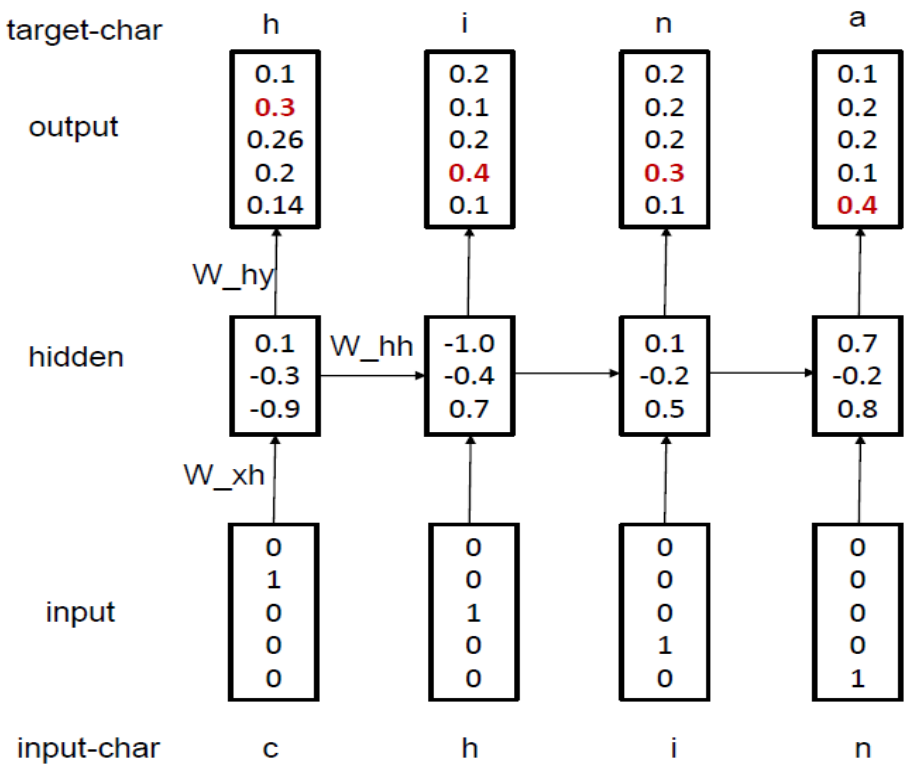
tanh([1.292, 1.392]) = [0.860, 0.884]

RNN的训练



- 在每个时间节点t = 0,1,2,3,4神经网络的输出都会产生误差值：E_0, E_1，E_2，E_3，E_4。与前馈神经网络类似，RNN也使用反向传播梯度下降法更新权重。

RNN的运行示例



RNN的前向传播案例

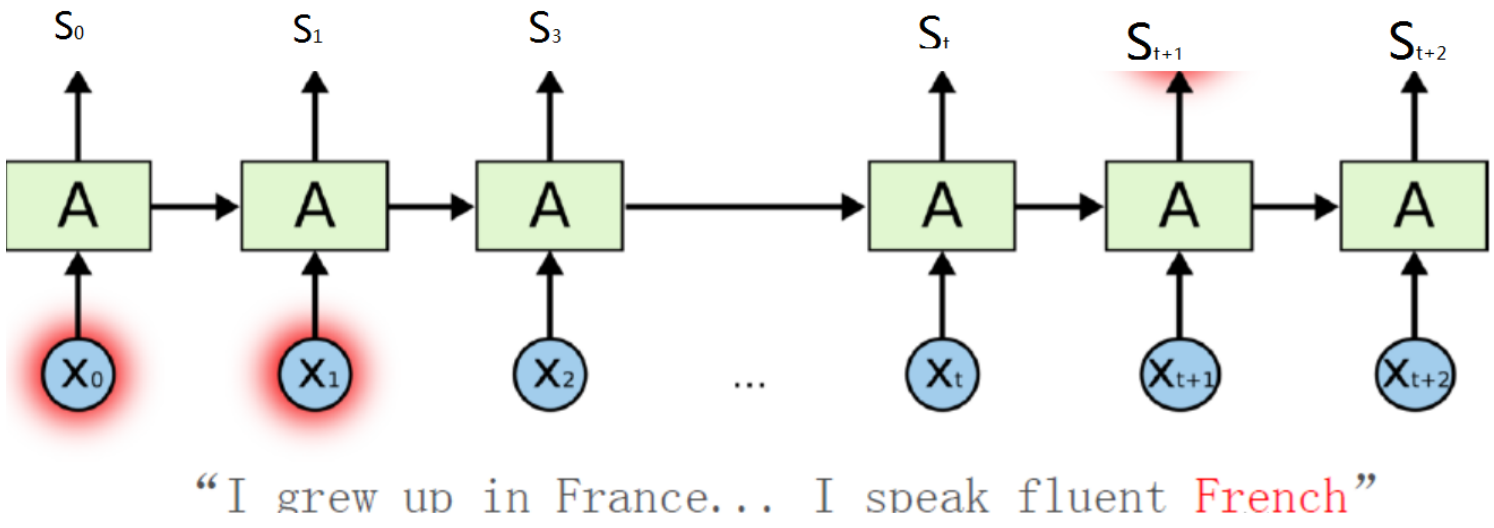
$$h_t = \tanh(w_{xh}x_t + w_{hh}h_{t-1})$$

$$y_t = softmax(w_{hy}h_t)$$

词向量空间：

a	c	h	l	n
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

RNN的不足

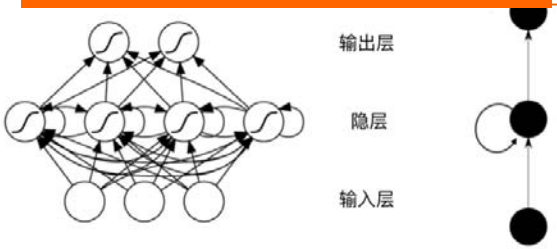


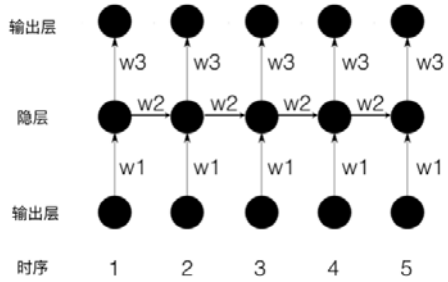
RNN缺陷：长期依赖(Long Term Dependencies)问题，产生长跨度依赖的问题。

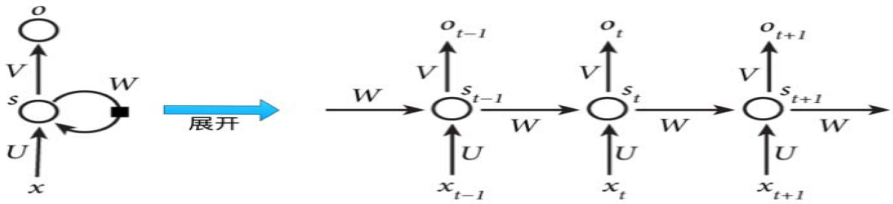
长短期记忆网络

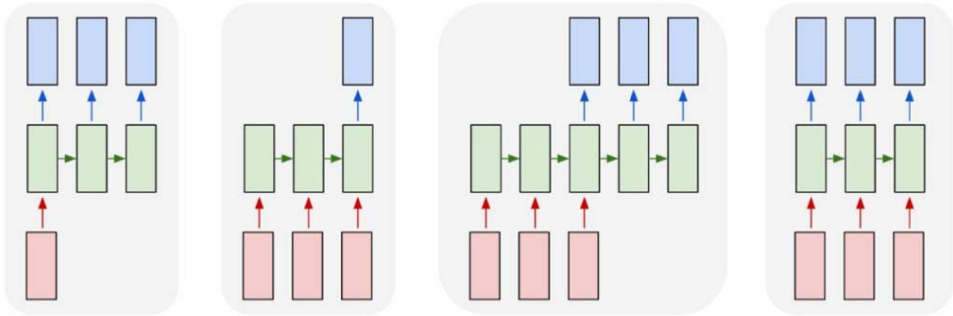
- 长短期记忆网络能够学习长期依赖关系，并可保留误差，在沿时间和层进行反向传递时，可以将误差保持在更加恒定的水平，让递归网络能够进行多个时间步的学习，从而建立远距离因果联系。它在许多问题上效果非常好，现在被广泛应用。

循环神经网络的类型









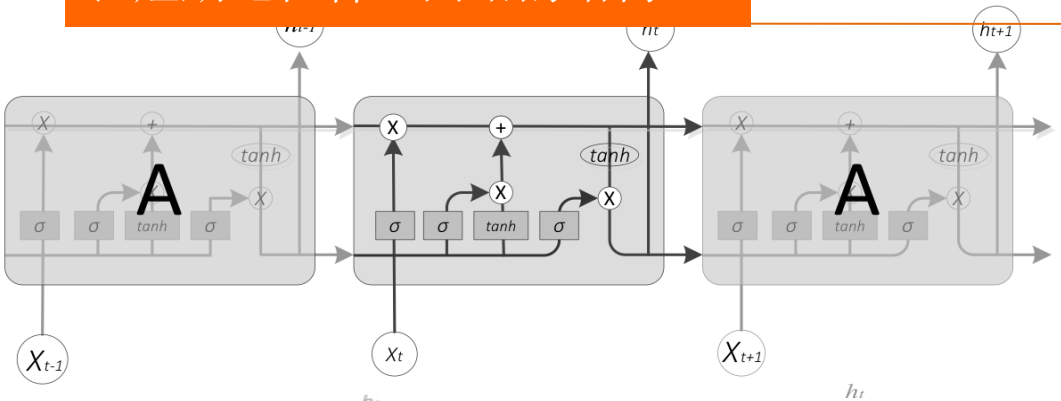
RNN分类：

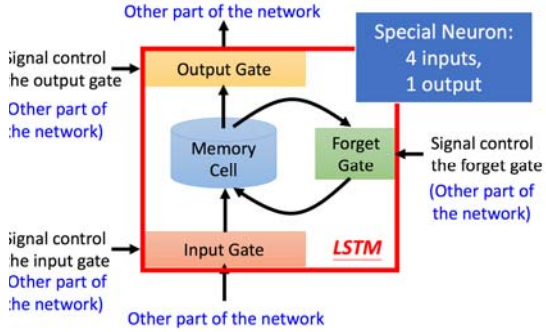
（1）输入一个输出多个，例如输入一张图像，输出这个图像的描述信息。
（2）输入是多个，输出则是一个，例如输入段话，输出这段话的情感。
（3）输入是多个，输出也是多个，如机器翻译输入一段话输出也是一段话（多个词）。
（4）多个输入和输出是同步的，例如进行字幕标记。

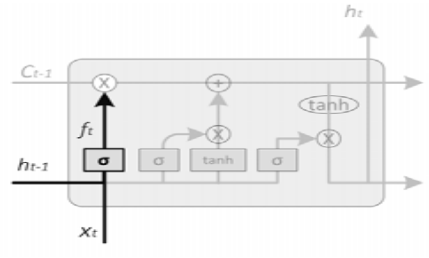
长短期记忆神经网络

- 长短期记忆网络将信息存放在递归网络正常信息流之外的门控单元中，这些单元可以存储、写入或读取信息，就像计算机内存中的数据一样。但愿通过门的开关判定存储哪些信息，何时允许读取、写入或清除信息。这些门是模拟的，包含输出范围全部在0～1之间的Sigmoid函数的逐元素相乘操作。这些门依据接收到的信号开关，而且会用自身的权重集对信息进行筛选，根据强度和输入内容决定是否允许信息通过。这些权重会通过递归网络的学习过程进行调整。

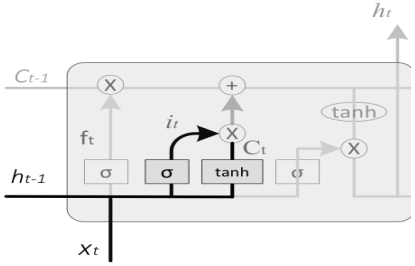
长短期记忆神经网络的结构



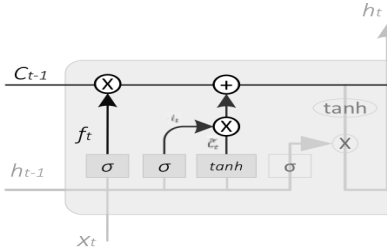




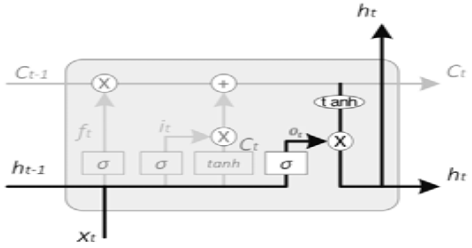
遗忘门的Sigmoid激活函数



一个新的候选值，判断是否将其加入到元胞状态中



保留重要信息，更新状态忘掉不重要信息



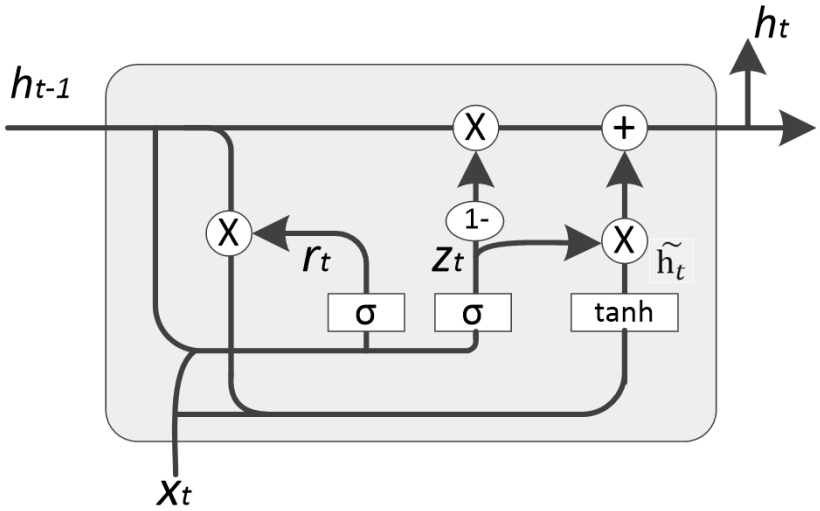
选择重要的信息作为元胞状态的输出

长短期记忆网络

- 长短期记忆网络的步骤如下：
  - 决定从元胞状态中扔掉哪些信息。由叫做“遗忘门”的Sigmoid层控制。遗忘门会输出0～1之间的数，1表示保留该信息，0表示丢弃该信息
  - 通过输入门将有用的新信息加入到元胞状态。首先，将前一状态和当前状态的输入输入到Sigmoid函数中滤除不重要信息。另外，通过tanh函数得到一个-1～1之间的输出结果。这将产生一个新的候选值，后续将判断是否将其加入到元胞状态中。
  - 将上一步中Sigmoid函数和tanh函数的输出结果相乘，并加上第一步中的输出结果，从而实现保留的信息都是重要信息，此时更新状态即可忘掉那些不重要的信息
  - 最后，从当前状态中选择重要的信息作为元胞状态的输出。首先，将前一隐状态和当前输入值通过Sigmoid函数得到一个0～1之间的结果值。然后对第三步中输出结果计算tanh函数的输出值，并与得到的结果值相乘，作为当前元胞隐状态的输出结果，同时也作为下一个隐状态的输入值

门限循环单元

- 门限循环单元本质上就是一个没有输出门的长短期记忆网络，因此它在每个时间步都会将记忆单元中的所有内容写入整体网络,其结构如下图所示。



门限循环单元

- 门限循环单元模型只有两个门，分别是更新门和重置门，更新门是遗忘门和输入门的结合体。将元胞状态和隐状态合并，更新门用于控制前一时刻的状态信息被带入到当前状态中的程度，更新门的值越大说明前一时刻的状态信息带入越多。重置门用于控制忽略前一时刻的状态信息的程度，重置门的值越小说明忽略的越多。这个模型比长短期记忆网络更加简化，也变得越来越流行。

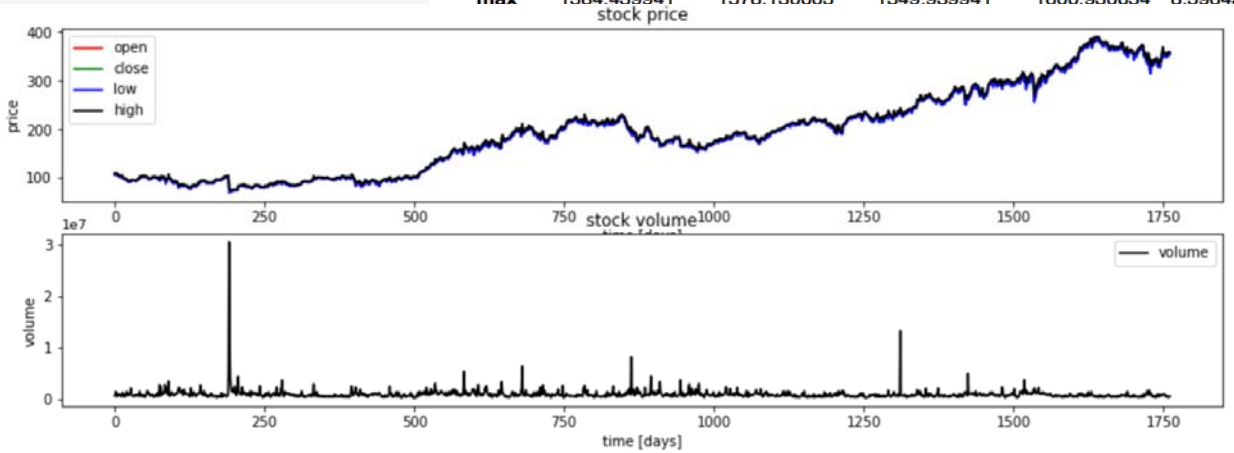
基于LSTM的股票预测-数据引入

date	symbol	open	close	low	high	volume
2016-12-30	ZBH	103.309998	103.199997	102.849998	103.930000	973800.0
2016-12-30	ZION	43.070000	43.040001	42.689999	43.310001	1938100.0
2016-12-30	ZTS	53.639999	53.529999	53.270000	53.740002	1701200.0
2016-12-30	AIV	44.730000	45.450001	44.410000	45.590000	1380900.0
2016-12-30	FTV	54.200001	53.630001	53.389999	54.480000	705100.0

	open	close	low	high	volume
count	851264.000000	851264.000000	851264.000000	851264.000000	8.512640e+05
mean	64.993618	65.011913	64.336541	65.639748	5.415113e+06
std	75.203893	75.201216	74.459518	75.906861	1.249468e+07
min	1.660000	1.590000	1.500000	1.810000	0.000000e+00
25%	31.270000	31.292776	30.940001	31.620001	1.221500e+06
50%	48.459999	48.480000	47.970001	48.959999	2.476250e+06
75%	75.120003	75.139999	74.400002	75.849998	5.222500e+06
max	1584.439941	1578.130005	1549.939941	1600.930054	8.596434e+08

指标数据：

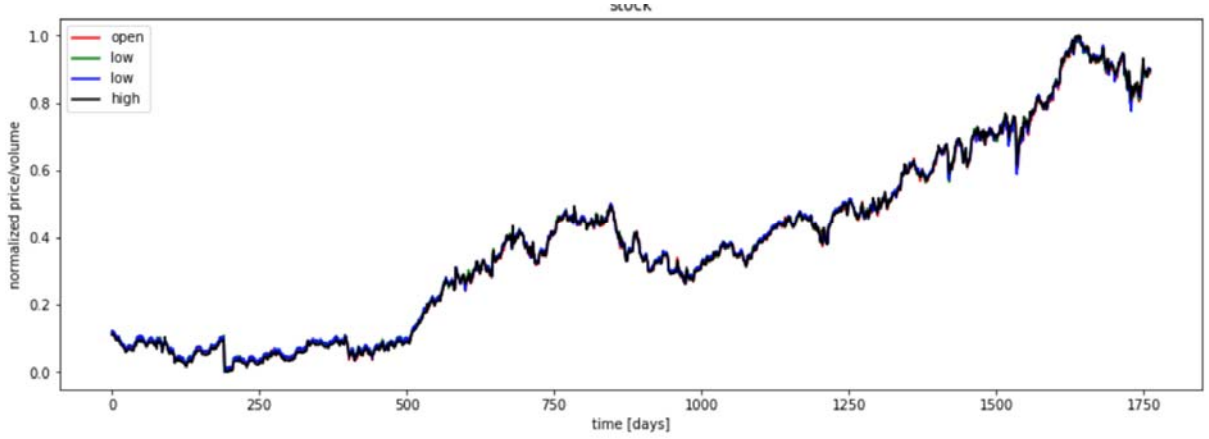
- 开盘
- 收盘
- 最低
- 最高
- 成交量



基于LSTM的股票预测-数据预处理

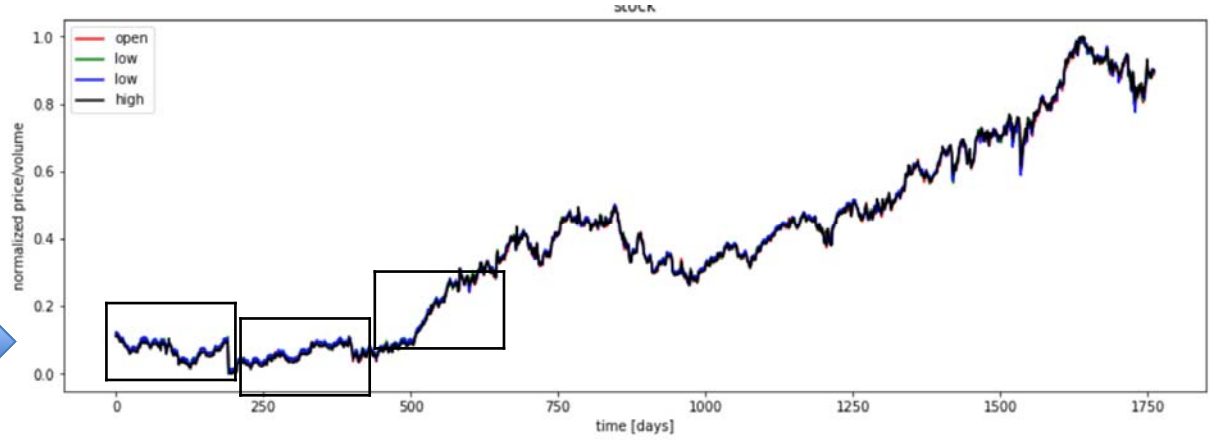
```python
# min-max 归一化
def normalize_data(df):
    min_max_scaler = sklearn.preprocessing.MinMaxScaler()
    df['open'] = min_max_scaler.fit_transform(df.open.values.reshape(-1,1))
    df['high'] = min_max_scaler.fit_transform(df.high.values.reshape(-1,1))
    df['low'] = min_max_scaler.fit_transform(df.low.values.reshape(-1,1))
    df['close'] = min_max_scaler.fit_transform(df['close'].values.reshape(-1,1))
    return df
```

数据预处理（归一化）
之后结果：



基于LSTM的股票预测-准备样本

加窗（window）



样本结果标记：窗口后涨跌作为样本标记,即窗口后的交易结果作为预测结果

模型采用MSE作为损失函数，对预测结果进行评价

股票走势转化为回归问题进行预测

基于LSTM的股票预测-设计LSTM网络

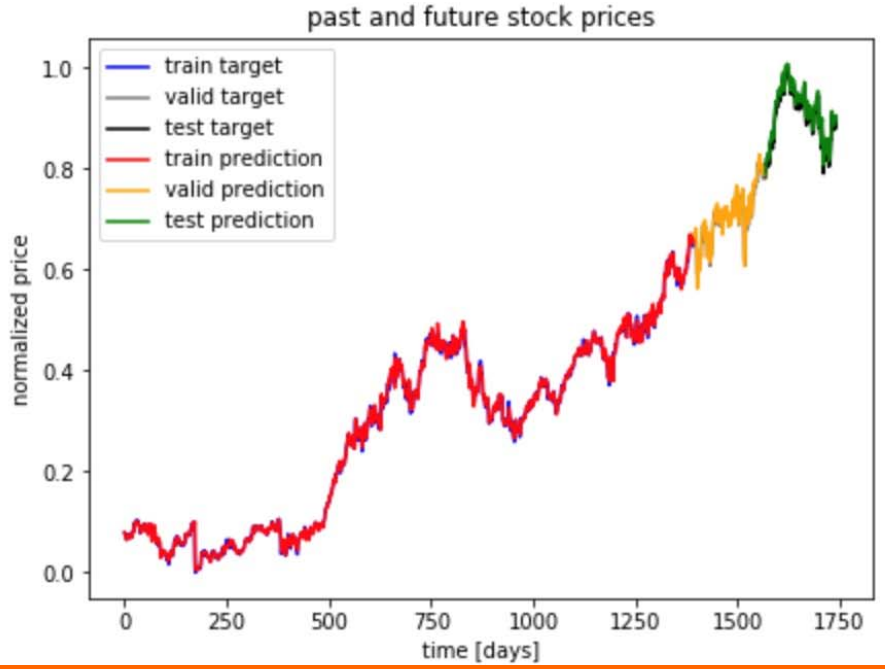
```
X = tf.placeholder(tf.float32, [None, n_steps, n_inputs])
y = tf.placeholder(tf.float32, [None, n_outputs])

layers = [tf.contrib.rnn.GRUCell(num_units=n_neurons, activation=tf.nn.leaky_relu)
          for layer in range(n_layers)]

multi_layer_cell = tf.contrib.rnn.MultiRNNCell(layers)
rnn_outputs, states = tf.nn.dynamic_rnn(multi_layer_cell, X, dtype=tf.float32)

stacked_rnn_outputs = tf.reshape(rnn_outputs, [-1, n_neurons])
stacked_outputs = tf.layers.dense(stacked_rnn_outputs, n_outputs)
outputs = tf.reshape(stacked_outputs, [-1, n_steps, n_outputs])
outputs = outputs[:,n_steps-1,:] # keep only last output of sequence

loss = tf.reduce_mean(tf.square(outputs - y)) # loss function = mean squared error
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)
training_op = optimizer.minimize(loss)
```

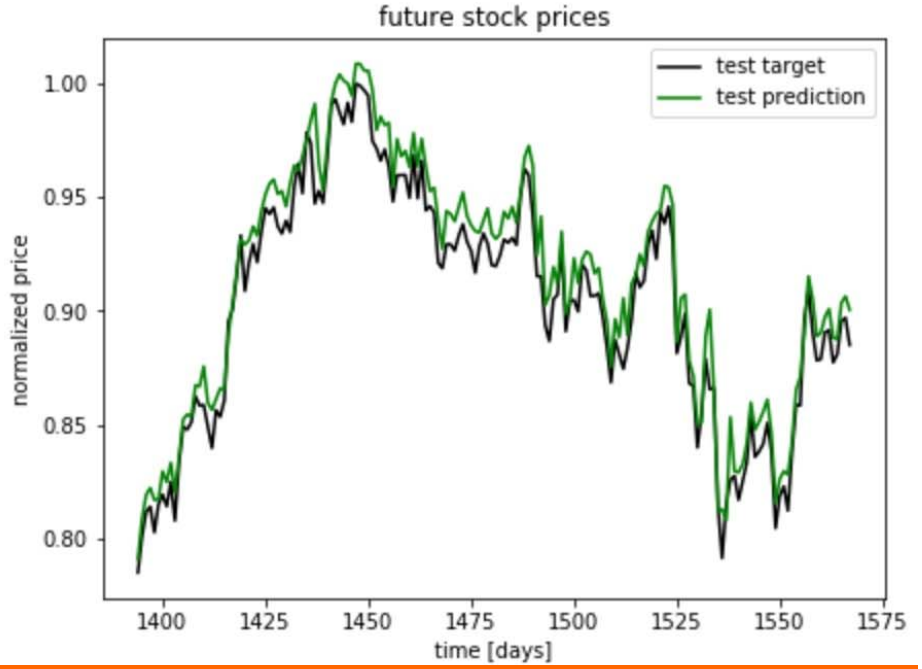
2层GRU（n_layers=2）

损失函数MSE

基于LSTM的股票预测-预测结果

- 训练epoch次数：10
- 训练集：1300+　验证集：170+　测试集170+





基于LSTM的电商秒杀业务预测

定义数据集：数据集包含通过订单量资源监控功能采集到的大量日志信息和实时监控系统采集到的CPU占用率，内存占用率，网络I/O状况。

- 数据以序列化表示，订单量资源监控功能采集数据的时间间隔为10秒。
- 每个序列样本单元由5个数据来表达。

序列化数据：

〈cpu_ratio，mem_ratio，net_io_port$_0$，net_io_port$_1$，net_io_port$_2$〉。

- cpu_ratio：cpu_ratio表示为150，这里50代表总的CPU占用率：(CPU使用百分比/系统CPU核数)，1代表是否有CPU核处于满负荷运转状态。
- mem_ratio：直接取使用已使用内存和服务器总物理内存的百分比数据。
- net_io_port$_i$：代表对某类TCP端口的I/O流量比率，选择80，8080，443和几个通用的数据库占用端口（例如oracle 1521，mysql 3306， sqlserver 1433），I/O比率居于前3位的放入数据采集样本中。

数据清洗：生成数据样本。

- 数据窗口为6小时。
- 以10秒为间隔，采集2160组数据作为一个窗口。
- 数据窗口样本：<s_0, s_1, … s_{2159}, s_{2160}>。
- 样本单元以单个整数来表达，即tokenizing。将样本状态标识为：正常和故障
- 在标注数据时以物理或云服务器宕机、挂起、应用异常、服务端口异常和I/O异常（不包含遭受网络攻击的状态）作为故障状态，其余状态均标记为正常状态。

s_i= cpu_ratio * 20% + mem_ratio * 30% + net_io_port$_0$ * 20% + net_io_port$_1$ * 15% + net_io_port$_2$ * 10%

基于LSTM的电商秒杀业务预测(1)

输入序列长度	2160，单个状态以整数表达，范围[0,120]
输出	状态0，1
LSTM层	2
隐含层cell数量	2000
损失函数计算	使用带权重的交叉熵来表示: $H(y,a)=-\frac{1}{n}\sum_{n=0}\sum_{i=0}y_{i,n}log(a_{i,n})$
梯度计算	$g_i=\sum_{j=0}^{len(y)}\frac{\partial y_j}{\partial x_i}$ $g=[g_0,g_1,\ldots,g_{len(x)}]$
梯度修剪	$L_t^i=\frac{L_t^i * N_c}{max(N_c,N_g)}$ $N_g=\sqrt{\sum_i (L_t^i)^2}$
激活函数	ReLU
优化参数	梯度下降优化
Dropout概率	输入：1.0输出0.8

- 为了分析这种海量文本型日志，采用2层LSTM来提供足够的信息表达能力，每层均包含2000个神经元来提供足够的记忆能力。
- 单个状态的整数表达范围为[0,120]，因此LSTM输入所需处理的token（表达类型）为121种。
- 梯度修正函数：

$$H(p,q)=-\frac{1}{N}\sum_x p(x)logq(x)$$

- 根据采集的训练样本，在虚拟机集群环境下的运行时间（训练建模时间）为8小时。每次建模所需迭代次数（控制收敛）为200次。
- 在TensorFlow中定义LSTM模型分为几个部分：定义变量，定义输入接口，循环执行LSTM Cell，定义loss，定义优化，定义预测。
- 循环运行LSTM Cell，展开LSTM

```
outputs = list()
output = saved_output
state = saved_state
for i in train_inputs:
    output, state = lstm_cell(i, output, state)
    outputs.append(output)
```

- 定义损失函数loss计算方法

```
with tf.control_dependencies([saved_output.assign(output),
 saved_state.assign(state)]):
logval = tf.nn.xw_plus_b(tf.concat(0, outputs), w, b)
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
                    logval, tf.concat(0, train_labels)))
```

基于LSTM的电商秒杀业务预测(2)

输入序列长度	2160，单个状态以整数表达，范围[0,120]
输出	状态0，1
LSTM层	2
隐含层cell数量	2000
损失函数计算	使用带权重的交叉熵来表示: $H(y,a)=-\frac{1}{n}\sum_{n=0}\sum_{i=0}y_{i,n}log(a_{i,n})$
梯度计算	$g_i=\sum_{j=0}^{len(y)}\frac{\partial y_j}{\partial x_i}$ $g=[g_0,g_1,\ldots,g_{len(x)}]$
梯度修剪	$L_t^i=\frac{L_t^i * N_c}{max(N_c,N_g)}$ $N_g=\sqrt{\sum_i (L_t^i)^2}$
激活函数	ReLU
优化参数	梯度下降优化
Dropout概率	输入：1.0输出0.8

- 为了分析这种海量文本型日志，采用2层LSTM来提供足够的信息表达能力，每层均包含2000个神经元来提供足够的记忆能力。
- 单个状态的整数表达范围为[0,120]，因此LSTM输入所需处理的token（表达类型）为121种。
- 梯度修正函数：

$$H(p,q)=-\frac{1}{N}\sum_x p(x)logq(x)$$

- 根据采集的训练样本，在虚拟机集群环境下的运行时间（训练建模时间）为8小时。每次建模所需迭代次数（控制收敛）为200次。
- 在TensorFlow中定义LSTM模型分为几个部分：定义变量，定义输入接口，循环执行LSTM Cell，定义loss，定义优化，定义预测。
- 循环运行LSTM Cell，展开LSTM

```
outputs = list()
output = saved_output
state = saved_state
for i in train_inputs:
    output, state = lstm_cell(i, output, state)
    outputs.append(output)
```

- 定义损失函数loss计算方法

```
with tf.control_dependencies([saved_output.assign(output),
 saved_state.assign(state)]):
logval = tf.nn.xw_plus_b(tf.concat(0, outputs), w, b)
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
                    logval, tf.concat(0, train_labels)))
```