Some notes about Embedding layer & LSTM

About Embedding layer:

- indexes -> vectors of fixed size

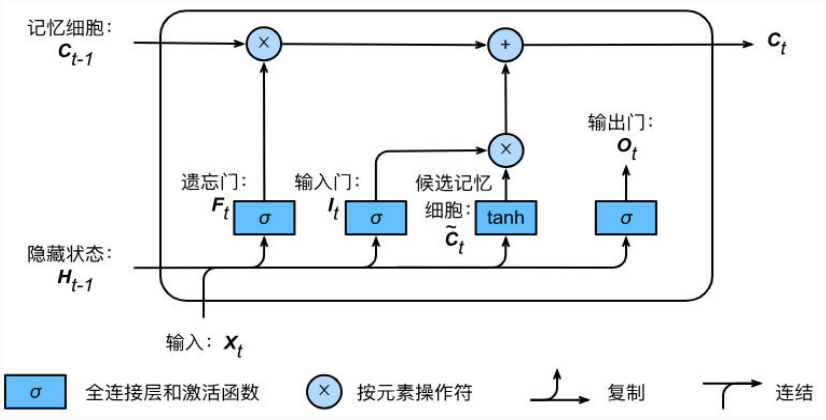
- input shape: (batch_size, seq_length)

- output shape: (batch_size, seq_length, output_dim)

- input dim: vocab_size

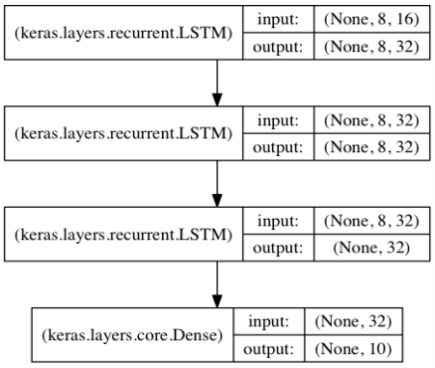
- input length: seq_length

Single LSTM cell:



长短期记忆中记忆细胞的计算。这里的乘号是按元素乘法。

Stacked-LSTM:




```
from keras.models import Sequential
from keras.layers import LSTM, Dense
import numpy as np

data_dim = 16
timesteps = 8
num_classes = 10

# expected input data shape: (batch_size, timesteps, data_dim)
model = Sequential()
model.add(LSTM(32, return_sequences=True,
               input_shape=(timesteps, data_dim)))  # returns a sequence of vectors of dimension 32
model.add(LSTM(32, return_sequences=True))  # returns a sequence of vectors of dimension 32
model.add(LSTM(32))  # return a single vector of dimension 32
model.add(Dense(10, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

About return_sequences=True:

45

You need to add `return_sequences=True` to the first layer so that its output tensor has `ndim=3` (i.e. batch size, timesteps, hidden state).

Please see the following example:

```
# expected input data shape: (batch_size, timesteps, data_dim)
model = Sequential()
model.add(LSTM(32, return_sequences=True,
               input_shape=(timesteps, data_dim)))  # returns a sequence of vectors of dimens
model.add(LSTM(32, return_sequences=True))  # returns a sequence of vectors of dimension 32
model.add(LSTM(32))  # return a single vector of dimension 32
model.add(Dense(10, activation='softmax'))
```

Note: timesteps is seq_length

Detailes of Stacked-LSTM: