【要点まとめ】
RNNの課題は、時系列を遡れば遡るほど勾配が消失してく。
ネットワークの構造自体を変えてそれを解決したのがLSTM。

LSTMで重要なのはコアとなるCEC。RNNから学習と記憶を分離し、記憶については全てここに任せる形をとる。
※つまりCECは記憶はできるが学習はできないため、NNであるためにはその周囲に学習するための入力ゲート・出力ゲートを配置してやる必要がある。

- 入力ゲート：「今回の入力値」「前回の出力値」を元に、最終的な今回の入力値をどうするか演算する
- 出力ゲート：「今回の出力」をどのように使うかを演算する
- 忘却ゲート：CECは基本的に与えられた情報は永劫記憶するので、適切なタイミングで不要な記憶を消去する

【実装演習】
ソースコードが無かったため、ゼロから作るDeepLearning②のLSTM実装を参考に

```
class LSTM:
    def __init__(self, Wx, Wh, b):
        self.params = [Wx, Wh, b]
        self.grads = [np.zeros_like(Wx), np.zeros_like(Wh),
np.zeros_like(b)]
        self.cache = None

    def forward(self, x, h_prev, c_prev):
        Wx, Wh, b = self.params
        N, H = h_prev.shape

        A = np.dot(x, Wx) + np.dot(h_prev, Wh) + b

        # slice
        f = A[:, :H]
        g = A[:, H:2*H]
        i = A[:, 2*H:3*H]
        o = A[:, 3*H:]

        f = sigmoid(f)
        g = np.tanh(g)
        i = sigmoid(i)
        o = sigmoid(o)

        c_next = f * c_prev + g * i
        h_next = o * np.tanh(c_next)

        self.cache = (x, h_prev, c_prev, i, f, g, o, c_next)
        return h_next, c_next
```

【実装演習考察】
CECの実装は記憶しておくべき値が多いので、それらを漏れなく意味も合わせて理解しておくことが重要だと感じました。
順伝播の実装自体は大変シンプルでわかりやすいです。（自己学習にも記述しましたが、tahnのお陰です。）

【自己学習】
※参考資料：ゼロから作るDeepLearning②
LSTMインタフェースの肝である(h(t-1)Wh + x(t)Wx +b)の計算が、numpyのtahnに隠されているため、 LSTMの実装はとてもシンプルに見える。
input、output、forgetは重要なゲートだが、forgetにより記憶を消去する場合は新しく覚えるべき情報も記憶セルに追加する必要がある。
その際にもtanhノードが使用される。