法律声明

- 课程详情请咨询
  - 微信公众号：小象
  - 新浪微博：ChinaHadoop

神经序列模型 II

主讲人： 史兴

07/07/2017

提纲

- 续：代码讲解
- Vanilla RNN
- LSTM
- RNNLM
- 应用展示

续：代码讲解

loss

cross-entropy

h5 b:0

softmax

h4 djdh4

output embedding djd_output_embed djd_output_embed_b

h3 djdh3

Tanh

h2 djdh2

linear transform djd_linear_w djd_linear_b

h1 djdh1

input embedding djd_input_embed

a:0

续：代码讲解

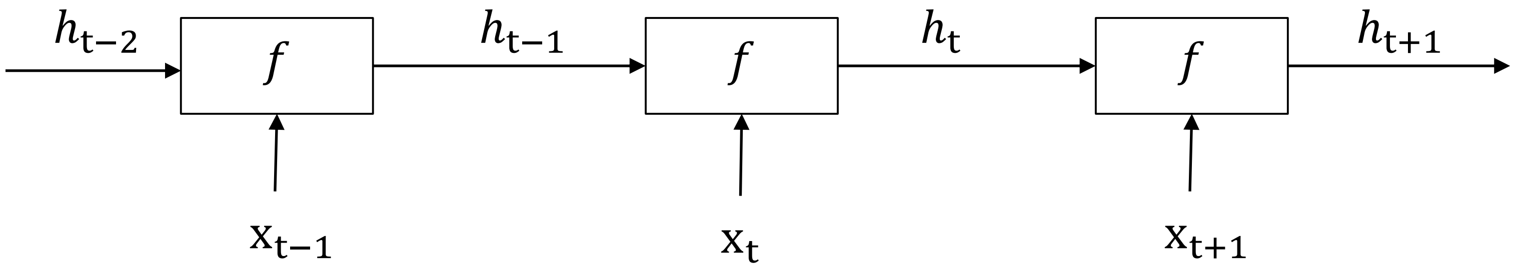
$y = f(x, \theta)$	$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial x}$	$\frac{\partial J}{\partial \theta} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial \theta}$
$y = xW + b$	$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial y} W^T$	$\frac{\partial J}{\partial W} = x^T \frac{\partial J}{\partial y}; \frac{\partial J}{\partial b} = \frac{\partial J}{\partial y}$
$y = \sigma(x)$	$\frac{\partial y}{\partial x} = y(1 - y)$	n/a
$y = tanh(x)$	$\frac{\partial y}{\partial x} = 1 - y^2$	n/a
$y = ReLU(x)$	$\frac{\partial y}{\partial x} = 1\ if\ x > 0; 0\ if\ x < 0$	n/a
$y = \text{ebl}\ (x)$	n/a	$\frac{\partial J}{\partial W[x,:]} = \frac{\partial J}{\partial y}$
$y = softmax\ (x)$ $J = CE\ (y, j)$	$\frac{\partial J}{\partial x_i} = y_i - 1, if\ i = j; y_i, if\ i \neq j$	n/a

续：代码讲解

- 转到github
  - Forward
  - Backward
  - Weight Update

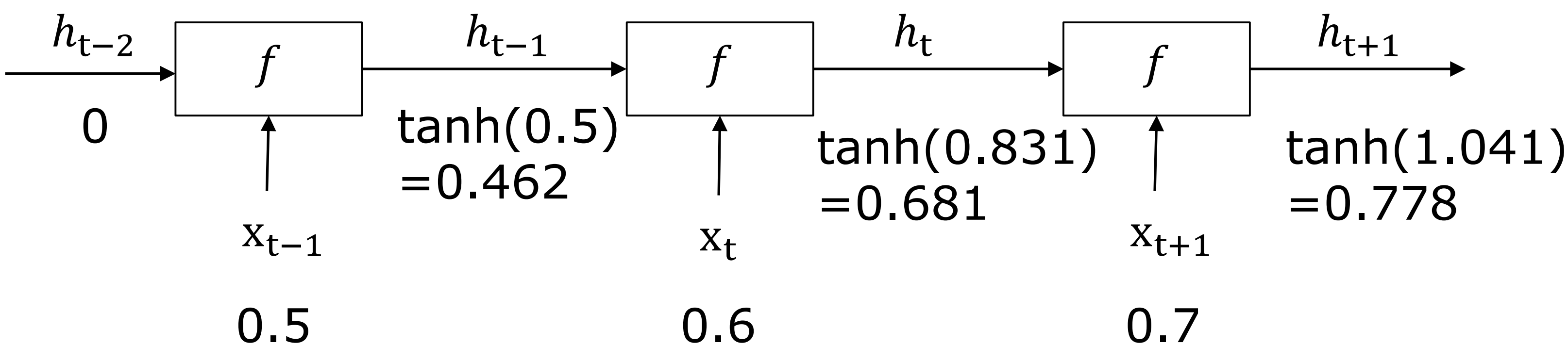
Vanilla RNN

- Recurrent Neural Network 循环神经网络
  - $h_t = f(h_{t-1}, x_t)$
  - 最简单的形式
    - $h_t = \tanh(h_{t-1}W + x_tU + b)$
    - W, U, b在每一步都是**一样**的



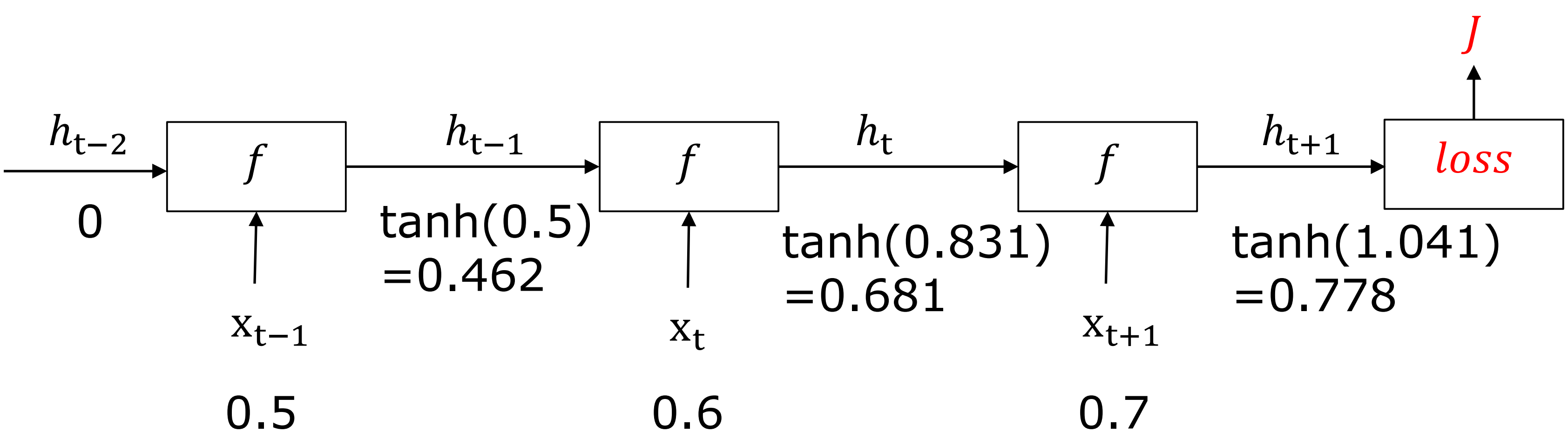
Vanilla RNN

- Recurrent Neural Network 循环神经网络
  - 例子
    - $h_t = \tanh(h_{t-1}W + x_tU + b)$
    - 为方便，x_t, h_t全部是一维向量
    - 参数：$W = 0.5, U = 1, b = 0$
    - Forward Propagation



Vanilla RNN

- Recurrent Neural Network 循环神经网络
  - 例子
    - $h_t = \tanh(h_{t-1}W + x_t U + b)$
    - 在结尾处增加一个loss function
    - 如何做back propagation去计算$\frac{\partial J}{\partial W}$?



Vanilla RNN

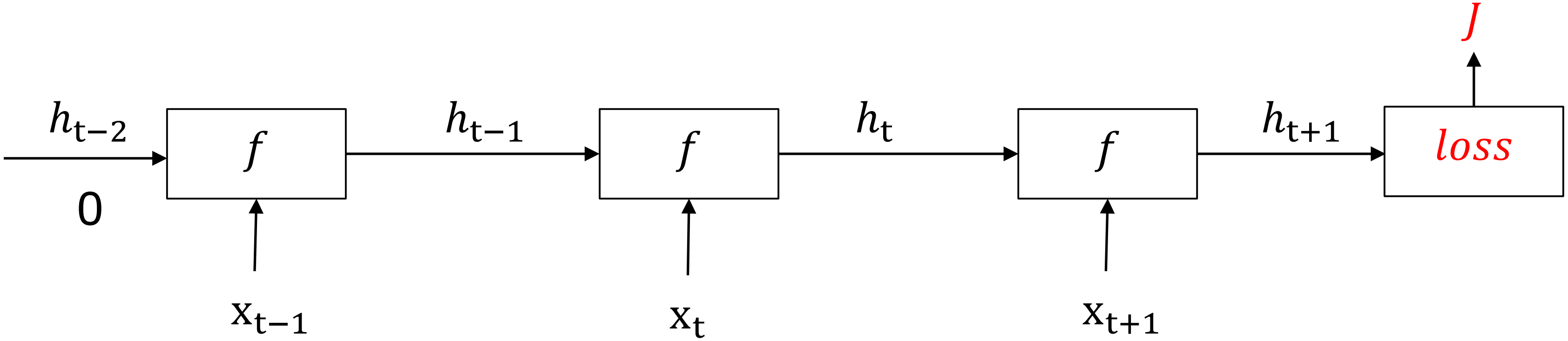
- Recurrent Neural Network 循环神经网络
  - $h_t = \tanh(h_{t-1}W + x_tU + b)$
  - 设$\mathrm{a}_t = h_{t-1}W + x_tU + b$
  - 假设只算两步：
    - $J = loss(h_{t+1}) = loss(\tanh(h_tW + x_{t+1}U + b))$
      $= loss(\tanh(\tanh(h_{t-1}W + x_tU + b)\,W + x_{t+1}U + b))$
  - $$\frac{\partial J}{\partial W} = \frac{\partial J}{\partial h_{t+1}}\frac{\partial \tanh(a_{t+1})}{\partial a_{t+1}}\left(h_t + W\frac{\partial h_t}{\partial W}\right)$$
    $$= \frac{\partial J}{\partial h_{t+1}}\frac{\partial \tanh(a_{t+1})}{\partial a_{t+1}}\left(h_t + W\frac{\partial \tanh(a_t)}{\partial a_t}h_{t-1}\right)$$
    $$= \frac{\partial J}{\partial h_{t+1}}\frac{\partial \tanh(a_{t+1})}{\partial a_{t+1}}h_t$$
    $$+ W\frac{\partial J}{\partial h_{t+1}}\frac{\partial \tanh(a_{t+1})}{\partial a_{t+1}}\frac{\partial \tanh(a_t)}{\partial a_t}h_{t-1}$$

Vanilla RNN

- Recurrent Neural Network 循环神经网络
  - $h_t = \tanh(h_{t-1}W + x_tU + b)$
  - 设$a_t = h_{t-1}W + x_tU + b$
  - 假设只算两步，将不同位置的W看做不同的w：
    - $J = loss(h_{t+1}) = loss(\tanh(h_t W_1 + x_{t+1}U + b))$
      $= loss(\tanh(\tanh(h_{t-1}W_2 + x_tU + b)\,W_1 + x_{t+1}U + b))$
  - $\frac{\partial J}{\partial W_1} = \frac{\partial J}{\partial h_{t+1}} \frac{\partial \tanh(a_{t+1})}{\partial a_{t+1}} h_t;$
  - $\frac{\partial J}{\partial W_2} = \frac{\partial J}{\partial h_{t+1}} \frac{\partial \tanh(a_{t+1})}{\partial a_{t+1}} \mathrm{w}_1 \frac{\partial \tanh(a_t)}{\partial a_t} h_{t-1};$
  - 所以，$\frac{\partial J}{\partial W} = \frac{\partial J}{\partial W_1} + \frac{\partial J}{\partial W_2}$

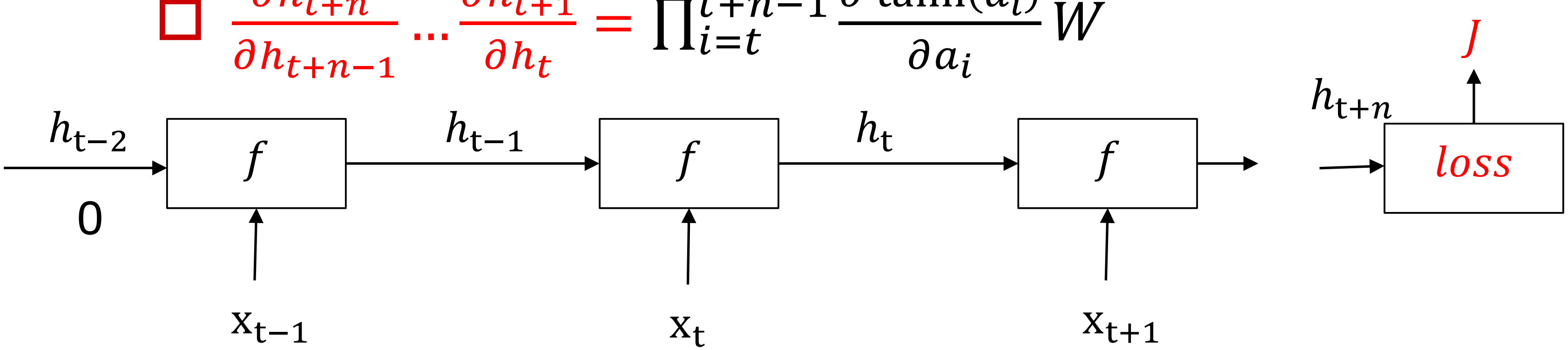
Vanilla RNN

- Recurrent Neural Network 循环神经网络
  - 例子
    - $h_t = \tanh(h_{t-1}W + x_tU + b)$
    - 在结尾处增加一个loss function
    - 如何做back propagation去计算$\frac{\partial J}{\partial W}$?
      - 不同位置的W看做是不同的W，分别计算偏导数后相加



Vanilla RNN

- Recurrent Neural Network 循环神经网络
  - $h_t = \tanh(h_{t-1}W + x_tU + b)$
  - $J = loss(h_{t+n})$
  - $\frac{\partial J}{\partial W_t} = \frac{\partial J}{\partial h_{t+n}} \frac{\partial h_{t+n}}{\partial h_{t+n-1}} \cdots \frac{\partial h_{t+1}}{\partial h_t} \frac{\partial h_t}{\partial W_t}$
  - $\frac{\partial h_{t+1}}{\partial h_t} = \frac{\partial \tanh(a_t)}{\partial a_t} W$
  - $\frac{\partial h_{t+n}}{\partial h_{t+n-1}} \cdots \frac{\partial h_{t+1}}{\partial h_t} = \prod_{i=t}^{t+n-1} \frac{\partial \tanh(a_i)}{\partial a_i} W$

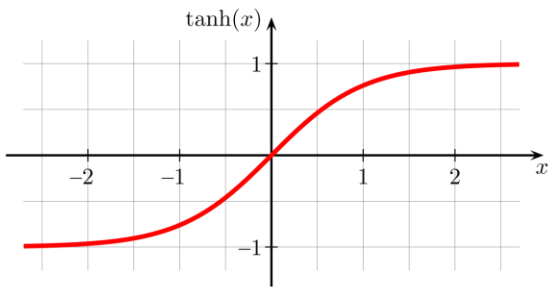


Vanilla RNN

- Recurrent Neural Network 循环神经网络
  - $\frac{\partial J}{\partial W_t} = \frac{\partial J}{\partial h_{t+n}} \frac{\partial h_{t+n}}{\partial h_{t+n-1}} \cdots \frac{\partial h_{t+1}}{\partial h_t} \frac{\partial h_t}{\partial W_t}$
  - $\frac{\partial h_{t+n}}{\partial h_{t+n-1}} \cdots \frac{\partial h_{t+1}}{\partial h_t} = \prod_{i=t}^{t+n-1} \frac{\partial \tanh(a_i)}{\partial a_i} W$
  - 假设W是一个一维的矩阵：
    - $|W| < 1,\ \prod_{i=t}^{t+n-1} W \to 0$
      - Vanishing Gradients (梯度消失)
    - $|W| > 1,\ \prod_{i=t}^{t+n-1} W \to \infty$
      - Exploding Gradients (梯度爆炸)
  - 如果W是高维的矩阵呢?
    - $|W| = \lambda_{max}(W)$, 最大的特征值

Vanilla RNN

- Recurrent Neural Network 循环神经网络
  - $\frac{\partial J}{\partial W_t} = \frac{\partial J}{\partial h_{t+n}} \frac{\partial h_{t+n}}{\partial h_{t+n-1}} \cdots \frac{\partial h_{t+1}}{\partial h_t} \frac{\partial h_t}{\partial W_t}$
  - $\frac{\partial h_{t+n}}{\partial h_{t+n-1}} \cdots \frac{\partial h_{t+1}}{\partial h_t} = \prod_{i=t}^{t+n-1} \frac{\partial \tanh(a_i)}{\partial a_i} W$
  - 考虑tanh(a)：
    - 多个$|a_i| > 2, \prod_{i=t}^{t+n-1} \frac{\partial \tanh(a_i)}{\partial a_i} \to 0$
      - Vanishing Gradients (梯度消失)



Vanilla RNN

- Recurrent Neural Network 循环神经网络
  - Exploding Gradients (梯度爆炸)

  - 解决方法： Gradient Clipping

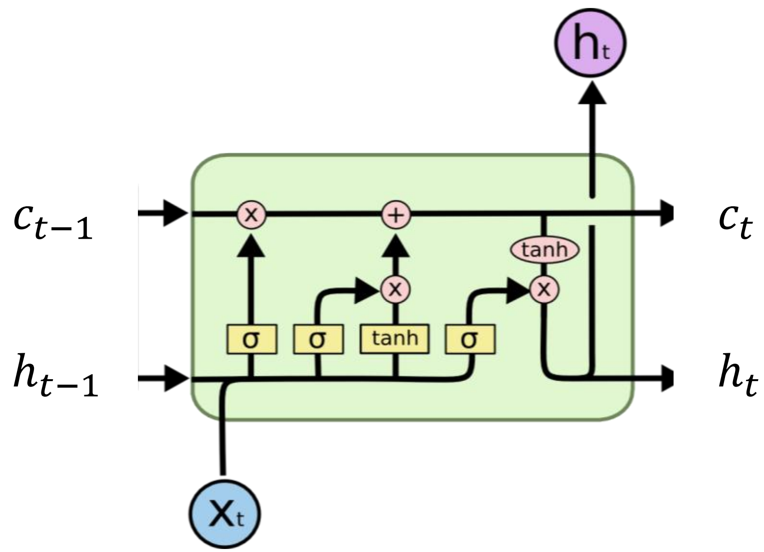
- $$\nabla = \begin{cases} \frac{c}{|\nabla|}\nabla, if\ |\nabla| > c; \\ \nabla, otherwise \end{cases}$$

Vanilla RNN

- Recurrent Neural Network 循环神经网络
  - Vanishing Gradients (梯度消失)
    - $\frac{\partial J}{\partial W_t} \to 0$, 远处的error所带来的指导意义消失了
  - 问题出在哪里?
    - $h_t = \tanh(h_{t-1}W + x_t U + b)$
    - $\frac{\partial h_{t+n}}{\partial h_{t+n-1}} \cdots \frac{\partial h_{t+1}}{\partial h_t} = \prod_{i=t}^{t+n-1} \frac{\partial \tanh(a_i)}{\partial a_i} W$
    - 递推公式本身有问题!
  - 解决方法:
    - LSTM !

LSTM

- Long Short-Term Memory (LSTM)
  - 尝试解决的问题：Vanishing Gradients
    - Vanilla RNN: $h_t = f(h_{t-1}, x_t)$
    - LSTM: $h_t, c_t = f(h_{t-1}, c_{t-1}, x_t)$

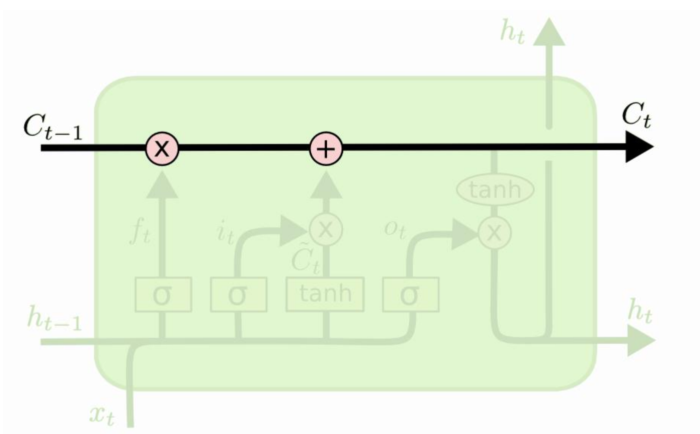


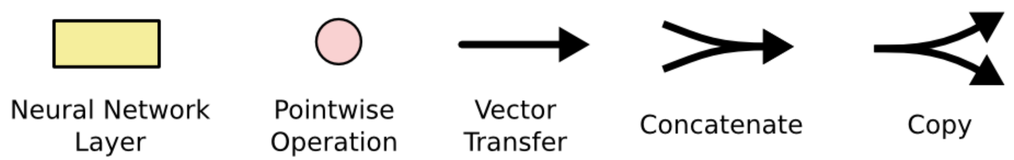
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM

- Long Short-Term Memory (LSTM)
  - 最关键的设计：C_t (cell state)
    - $C_t = C_{t-1} * \mathrm{f} + i$ 全部都是线性操作
      - $*$，$+$：elementwise
      - f: [0,1]，忘记的程度

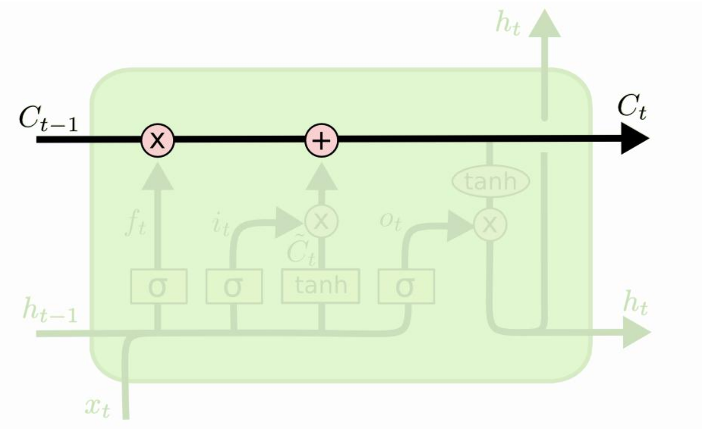
c_{t-1}	0.5	-0.2	0.5
f	1	0	0.5
$C_{t-1} * \mathrm{f}$	0.5	0	0.25
i	-0.5	0.5	2
c_t	0	0.5	2.25





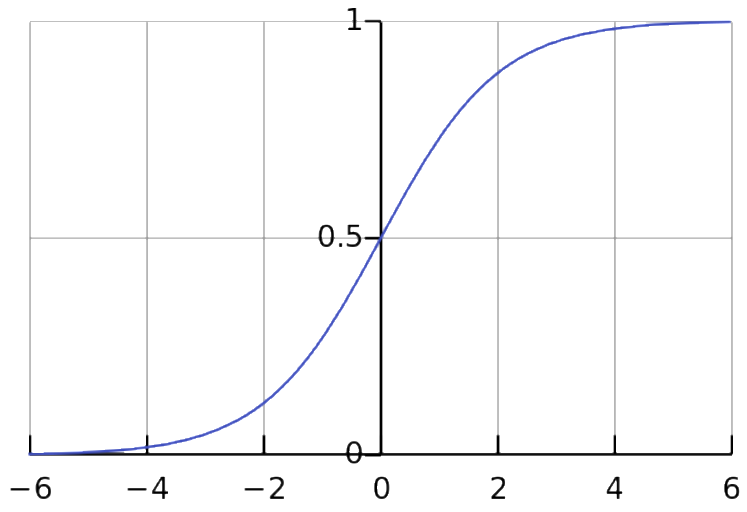
LSTM

- Long Short-Term Memory (LSTM)
  - 最关键的设计：C_t (cell state)
    - $C_t = C_{t-1} * \mathrm{f} + i$ 全部都是线性操作
      - *，+：elementwise
      - f: [0,1]，忘记的程度
    - 对比Vanilla RNN
      - $\frac{\partial h_{t+1}}{\partial h_t} = \frac{\partial \tanh(a_t)}{\partial a_t} W$
      - $\frac{\partial c_{t+1}}{\partial c_t} = \mathrm{f}$
      - 在不同的位置t上的f是不同的，所以做到连续多个f的值接近1，这样对应的gradient就不会趋近于0



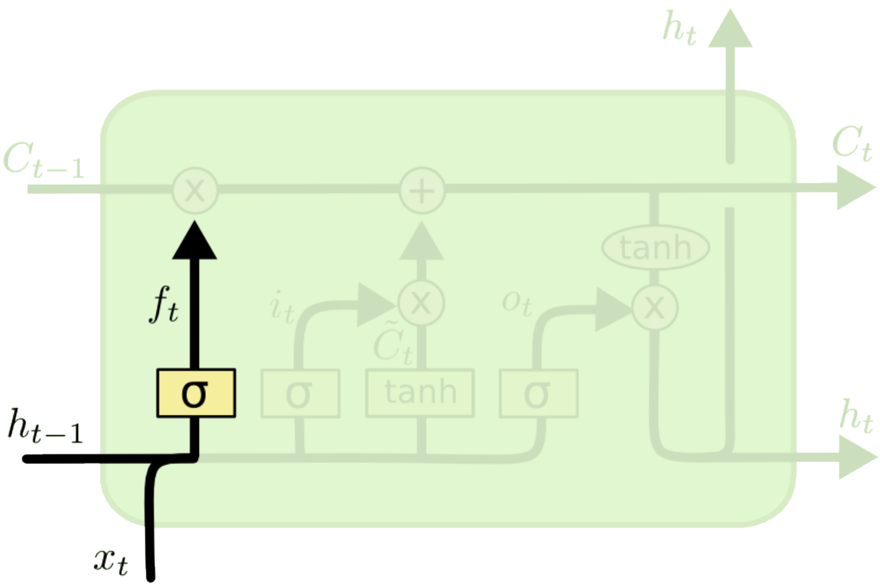
神经网络基础

- 非线性变换：sigmoid
  - $y = \sigma(x) = \frac{1}{1+e^{-x}}$
  - $\frac{\partial y}{\partial x} = y(1-y)$
  - 值域[0,1]
    - 做概率
    - 做开关
    - 做“挤压”
  - 导数的绝对值
    - 只在0附近比较大
  - elementwise



LSTM

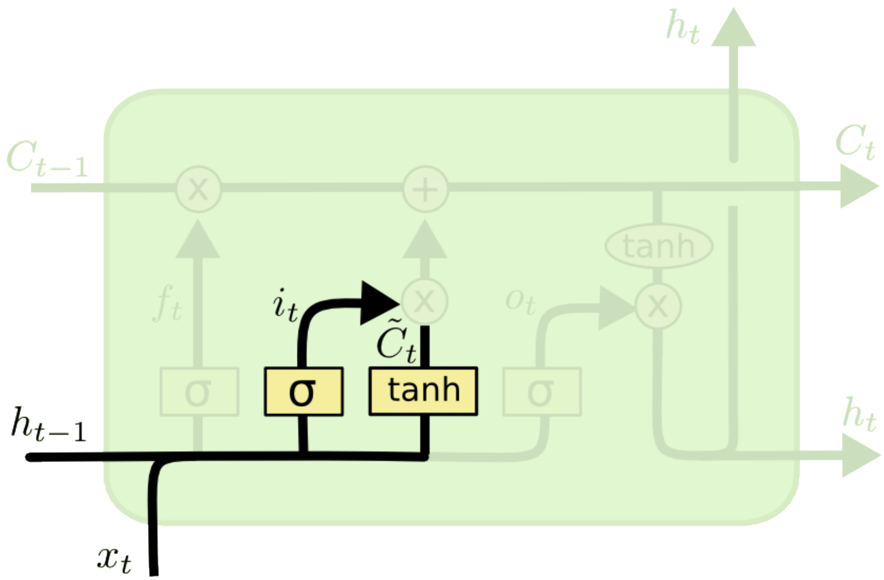
- Long Short-Term Memory (LSTM)
  - forget gate



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

LSTM

- Long Short-Term Memory (LSTM)
  - input gate 和 input

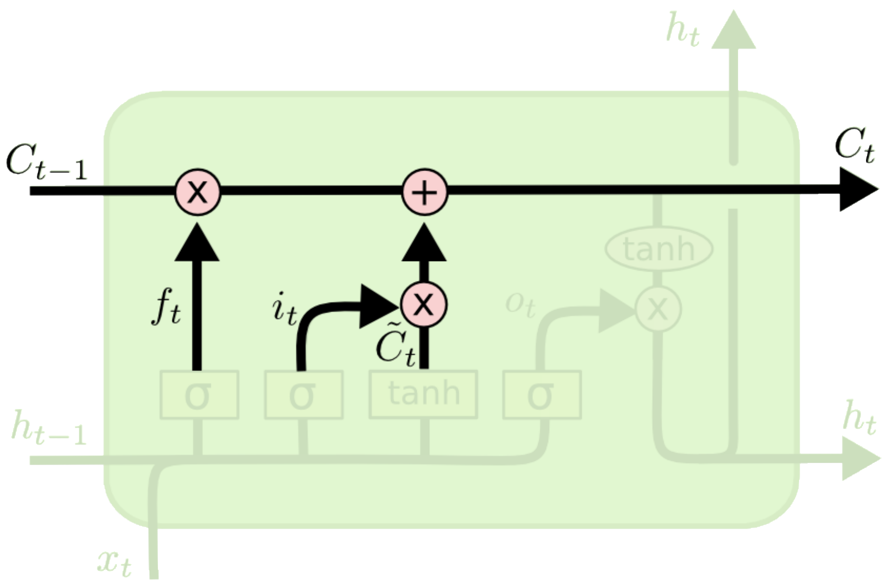


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

LSTM

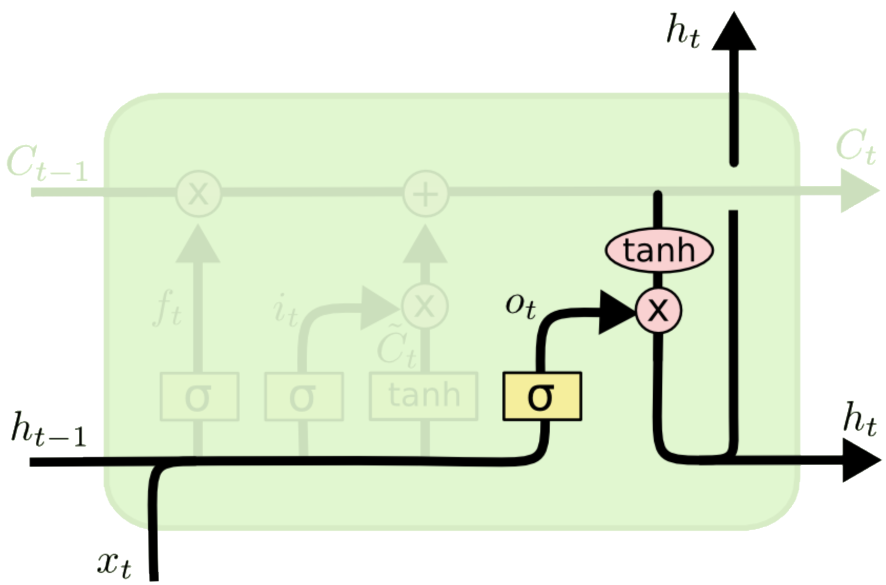
- Long Short-Term Memory (LSTM)
  - Cell State



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM

- Long Short-Term Memory (LSTM)
  - output gate 和 hidden state



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

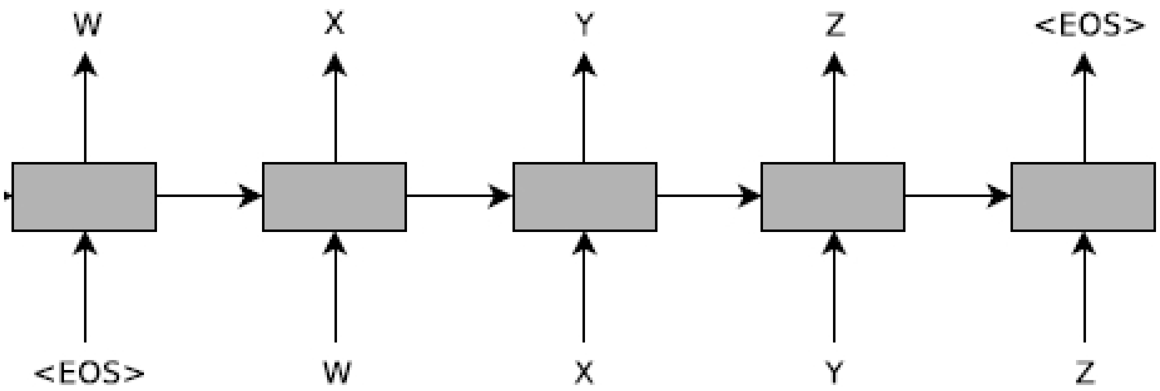
$$h_t = o_t * \tanh\left(C_t\right)$$

RNNLM

- n-gram 潜在的问题:
  - P(梨|水果 包含) P(苹果|水果 包含)
    - 对“词”的理解有限
      - Neural Network Language Model
    - N-gram 上下文的长度有限
      - Recurrent NN Language Model

RNNLM

句子：”w x y z”



~~N-gram 上下文的长度有限~~

应用展示

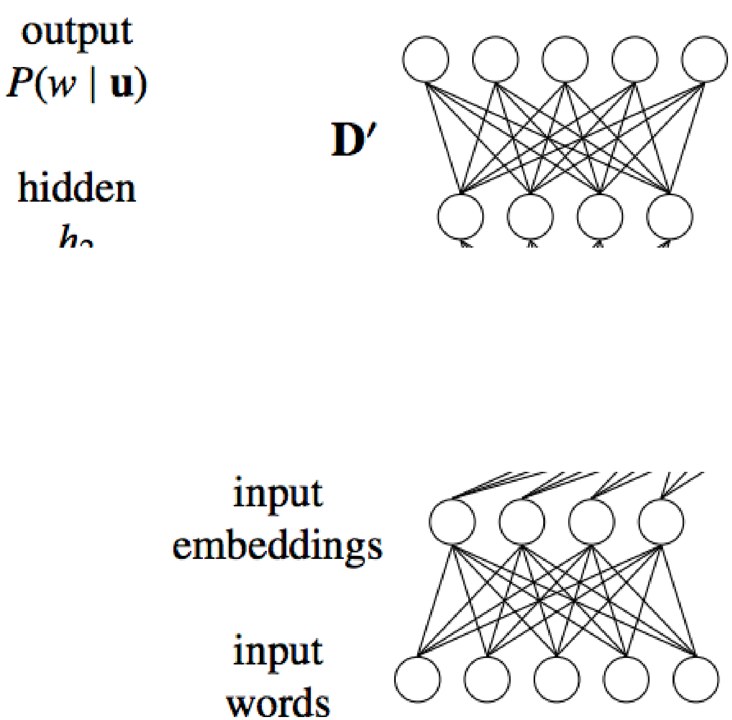
- 应用展示
  - 在tensorflow中实现LSTM Language Model
  - 代码地址：
    - https://github.com/shixing/xing_nlp/tree/master/LM/RNNLM
  - 本次要求：
    - 只需要大家可以跑通就好
    - 有时间的，可以预习一下
  - 下节课预告：
    - 本次代码详细讲解

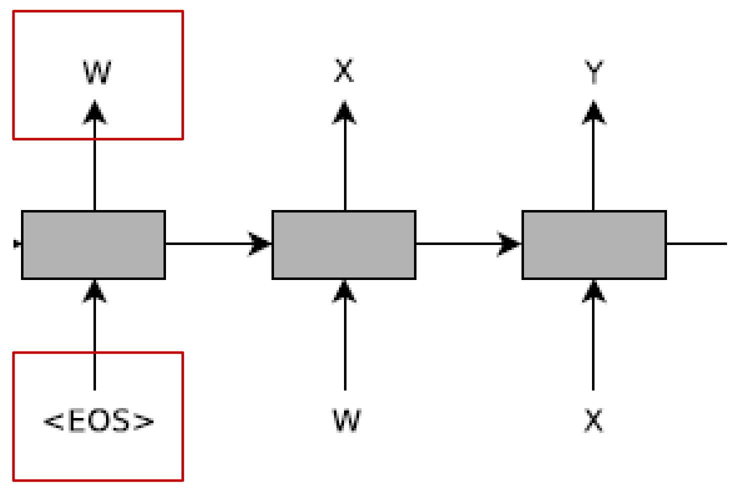
应用展示

- 应用展示
  - 运行代码：
    - cd sh
    - 小模型，小数据
      - bash train_small.sh
    - 大模型，大数据(需要GPU)
      - bash train_ptb.sh

RNNLM

Softmax Layer + Cross-entropy Loss



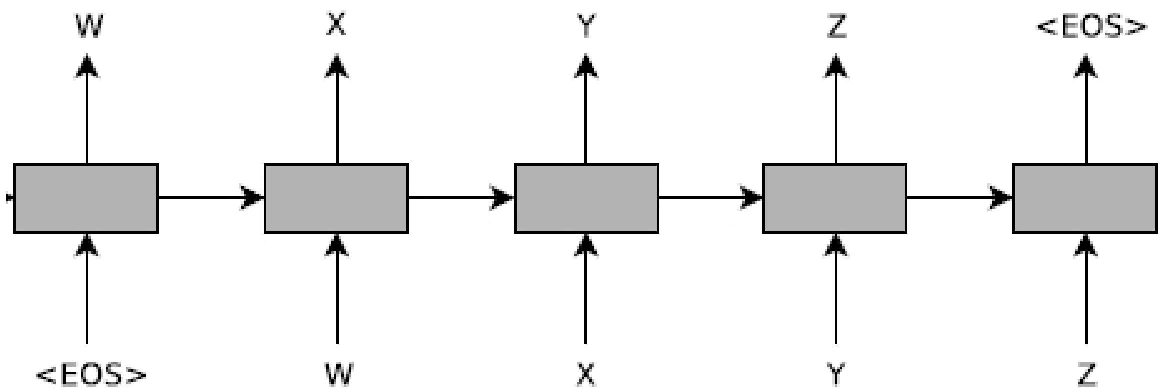


Embedding Lookup

RNNLM

句子：”w x y z”

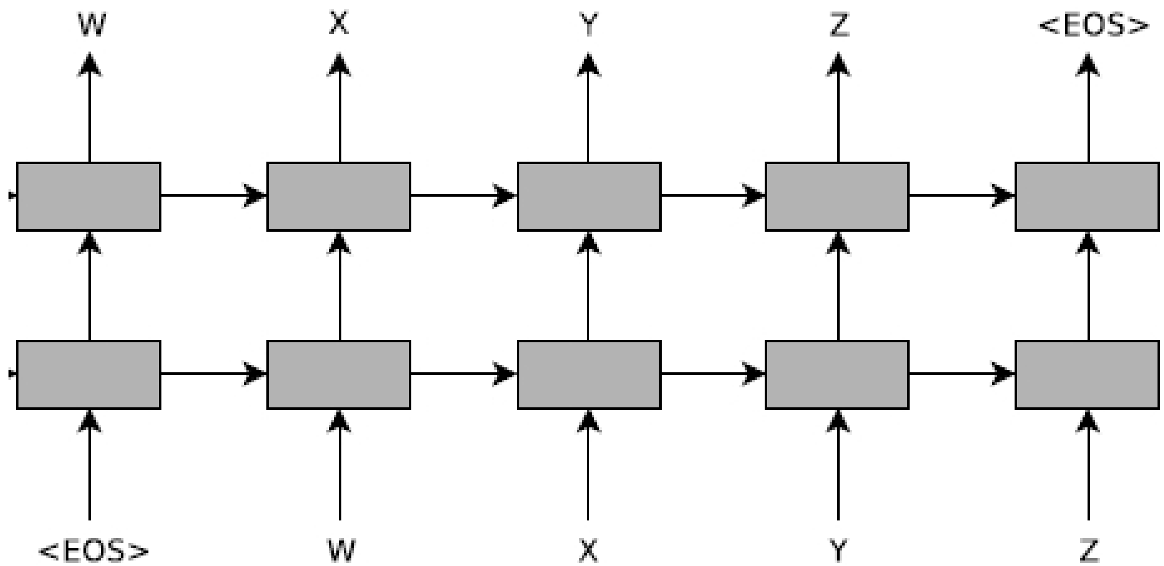
Loss = loss(w) + loss(x) + loss(y) + loss(z) + loss(<EOS>)



RNNLM

多层叠加

每层的参数不一样



RNNLM

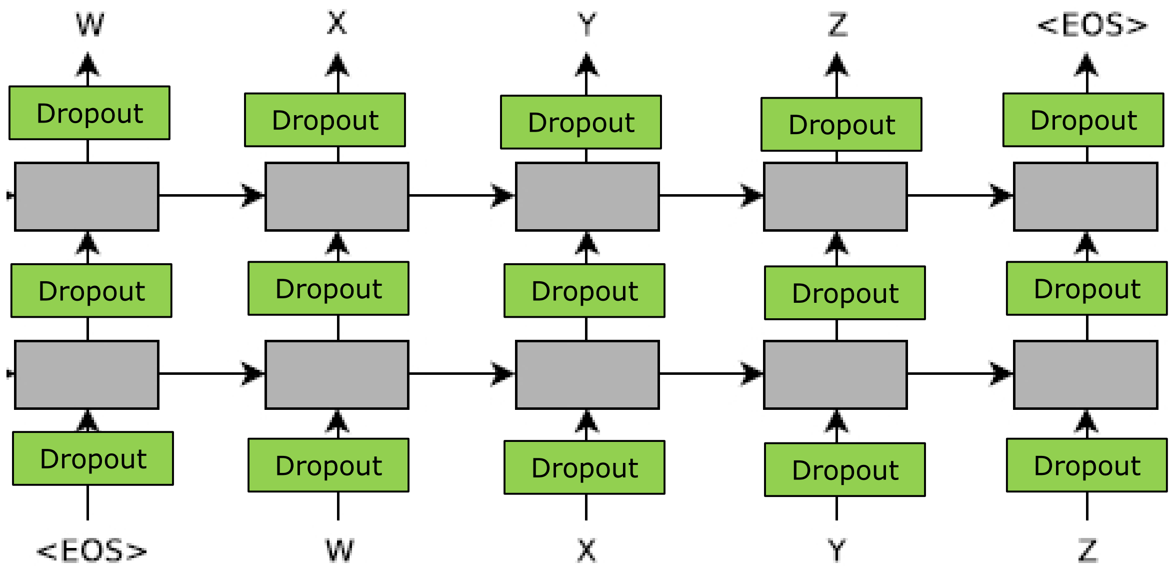
- Regulation： Dropout
  - $y = dropout(x, r)$
  - $r \in [0,1]$: dropout rate
  - $x \in R^d$
  - $y_i = \begin{cases} \frac{x_i}{r}, if\ rand \leq r \\ 0, if\ rand > r \end{cases}$

r = 0.5

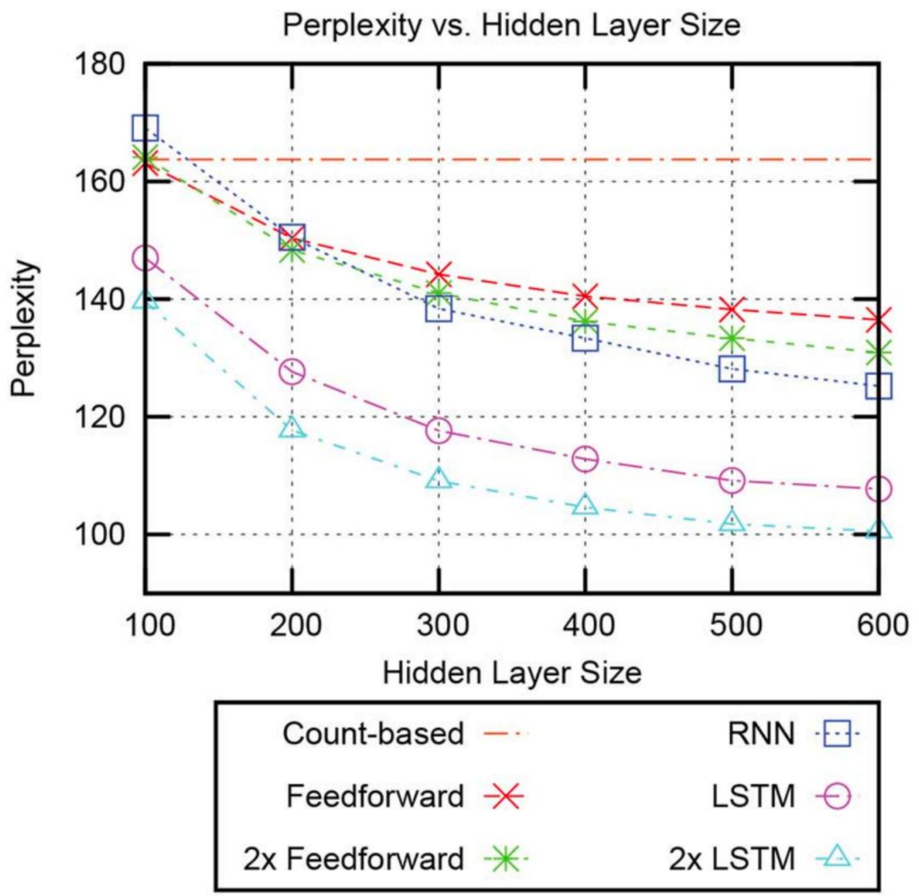
x	0.5	0.4	-0.5
rand	0.9	0.2	0.4
y	0	0.8	-1.0

RNNLM

多层叠加 + Dropout



RNNLM



(from Sundermeyer, Ney, Schlüter, IEEE TASLP 2015)

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：大数据分析挖掘
- 新浪微博：ChinaHadoop