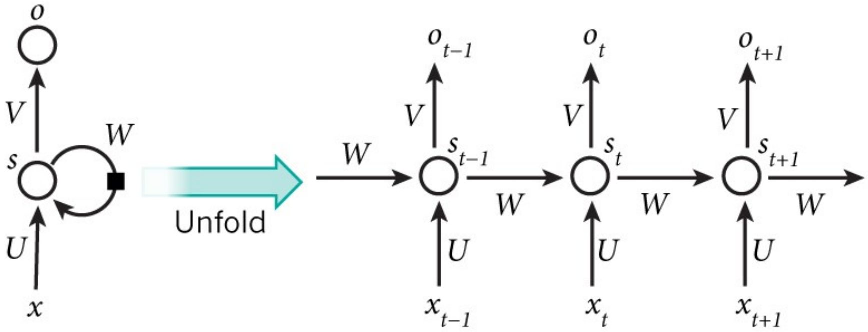
An Introduction to RNN and LSTM

Xueying Situ

November 5, 2017

Outline

RNN



Why Are They Called RNNs

- they perform the same task for every element of a sequence
- their output is depended on the previous computations

Features

- they have a "memory" which captures information about what has been calculated so far
- they can be seen as multiple layers neural network

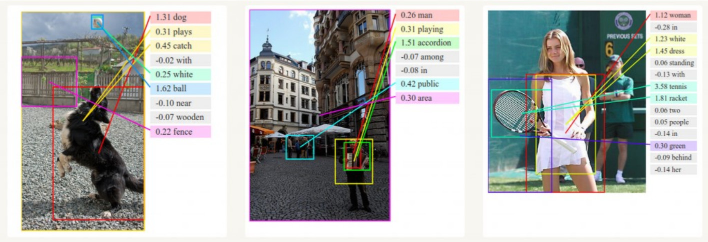
Formulas

$$s_t = f(Ux_t + Ws_t - 1)$$
$$o_t = soft\max(Vs_t)$$

- X$_t$ is the input at time step.A one-hot vector corresponding to the t+1th word of a sentence
- S$_t$ is the hidden state at time step t. It' s the "memory" of the network.
- O$_t$ is the output at step.A vector of probabilities across our vocabulary
- The function f usually is a nonlinearity such as tanh or ReLU.

Application

- Language Modeling and Generating Text
- Machine Translation
- Speech Recognition
- Generating Image Descriptions



Language Model

Let' s say we have sentence of words. A language model allows us to predict the probability of observing the?sentence (in a given dataset) as:

$$P(w_1, ..., w_m) = \prod_{i=1}^{m} P(w_i | w_1, ..., w_{i-1})$$

- Assumption: Word in a sentence depends on its previous words.
- Application
  - used as a scoring mechanism
  - generate new text

A Practice Example: RNN Used as Language Model

Goal: predict the next word conditioned on all previous words.

```
x: SENTENCE_START what are n't you understanding about this ? !
[0, 51, 27, 16, 10, 856, 53, 25, 34, 69]

y: what are n't you understanding about this ? ! SENTENCE_END
[51, 27, 16, 10, 856, 53, 25, 34, 69, 1]
```

x is a sequence of words, a matrix.

x_t is a single word. Each word is a one-hot vector of size vocabulary-size.

o is a sequence of words, a matrix.

o_t is a vector of vocabulary-size elements.Each element represents the probability of that word being the next word in the sentence.

BPTT

- RNN formula

$$s_t = f(Ux_t + Ws_t - 1)$$
$$o_t = soft\max(Vs_t)$$

- Loss function

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$
$$E(y_t, \hat{y}_t) = \sum_t E_t(y_t, \hat{y}_t) = -\sum_t y_t \log \hat{y}_t$$

- For $V = V - a\frac{\partial E_3}{\partial V}$

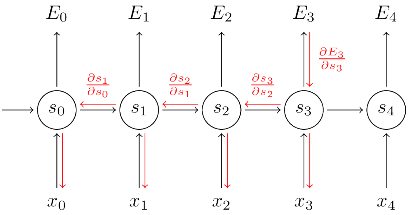
$$\frac{\partial E_3}{\partial V} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial z_3}\frac{\partial z_3}{\partial V} = (\hat{y}_3 - y_3) \otimes s_3 \qquad where \qquad z_3 = Vs_3$$

$$\frac{\partial E_3}{\partial V} = f(\hat{y}_3, y_3, s_3)$$

AN INTODUCTION TO RNNS

- For $W = W - a\frac{\partial E_3}{\partial W}$

$$\begin{aligned}\frac{\partial E_3}{\partial W} &= \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial W} \\ &= \sum_{k=0}^{3}\frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial s_k}\frac{\partial s_k}{\partial W}\end{aligned}$$



$$\frac{\partial s_3}{\partial s_k} = \prod_{k \leq i \leq 3} \frac{\partial s_i}{\partial s_{i-1}} = \prod_{k \leq i \leq 3} W^T diag(\tanh'(s_{i-1}))$$

$$\text{as} \quad \tanh' \leq 1, \text{if} \quad \|W\| < 1,$$

$$\left\| \frac{\partial s_i}{\partial s_{i-1}} \right\| = \left\| W^T \right\| \left\| diag(\tanh'(s_{i-1})) \right\| < 1$$

$$\frac{\partial s_3}{\partial s_k} \leq \eta^{3-k} \quad \text{where} \quad \eta < 1$$

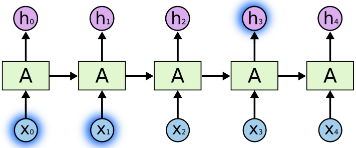
if $\frac{\partial s_3}{\partial s_k} \geq \eta^{3-k} \quad \text{where} \quad \eta > 1$

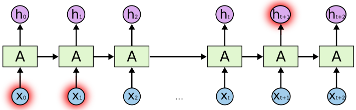
then $\|W\| > 1$

Exploding/Vanishing Gradient

- Vanishing Gradient
  the gradient values are shrinking?exponentially fast,hardly learn anything
- Exploding Gradient
  Get NAN error,the program will crash

RNNs's Limitation

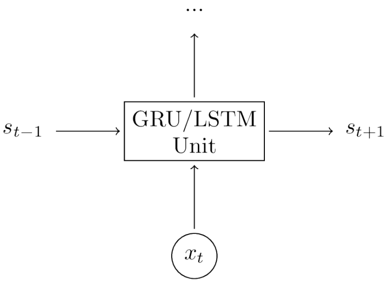




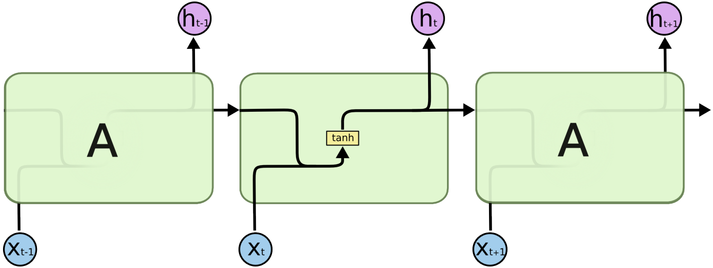
Solutions

- Solutions to exploding gradient
  - gradient clipping
- Solutions to vanishing gradient
  - clipping the gradients at a pre-defined threshold
  - use ReLU instead of tanh as activation function
  - LSTM/GRU

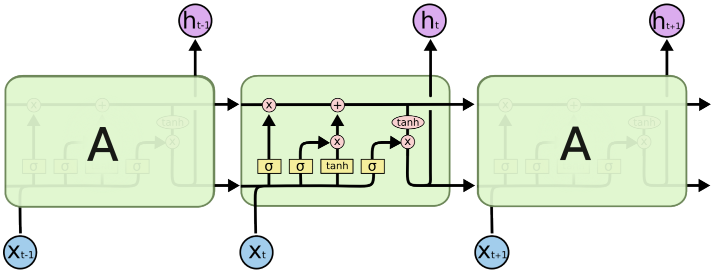
Long Short-term Memory



RNNs's Cell State

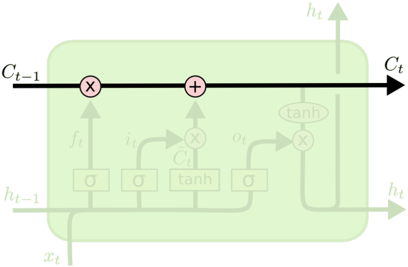


LSTMs's Cell State

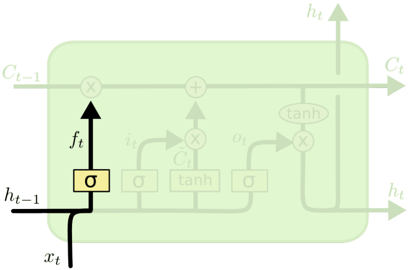


The Core Idea Behind LSTMs

- Cell State

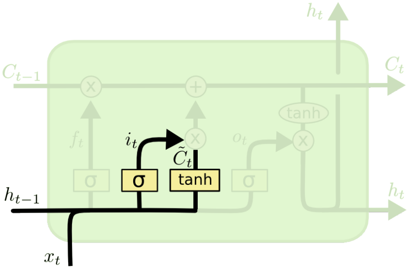


- Forget Gate



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

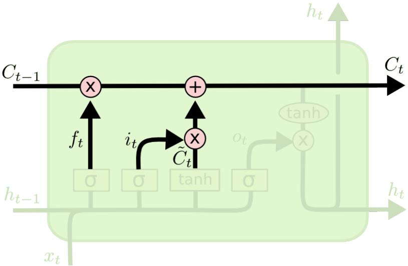
- Input Gate



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

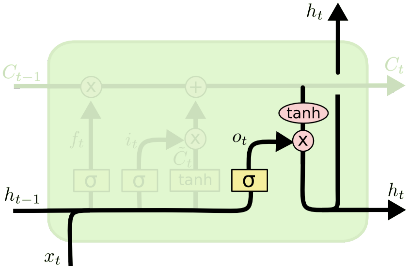
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

- Cell State



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

- Output Gate



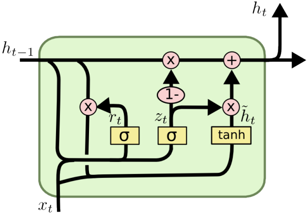
$$o_t = \sigma\left(W_o\ [\,h_{t-1}, x_t]\ +\ b_o\right)$$

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

Why LSTMs Can Solve Gradient Vanish in RNNs

GRU

- two gates: update gate(combine f and i) and reset gate
- no cell state
- without a second nonlinearity when computing the output



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Citations

1. Pascanu R, Mikolov T, Bengio Y. On the difficulty of training Recurrent Neural Networks[J]. Computer Science, 2012, 52(3):III-1310.
2. Recurrent Neural Networks Tutorial
3. Understanding LSTM Networks

Questions?