GATES, GRU

Deep Neural Networks

Session 19

Pramod Sharma
pramod.sharma@prasami.com

2

Agenda

- Where we are
- Vanishing and exploding gradients
- Keeping Things Stable
- GRU Units

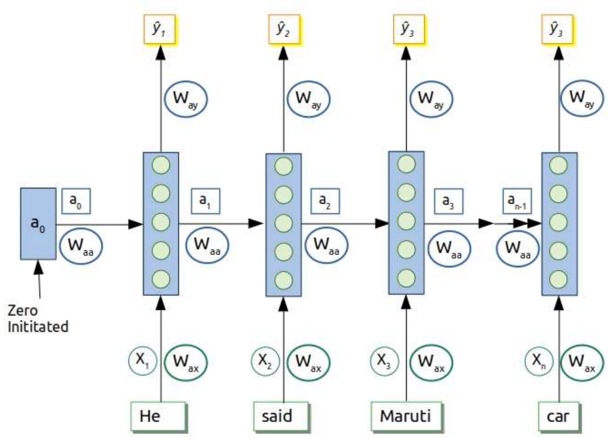
5/25/2024

pra-sâmi

Recurrent Neural Networks (RNNs)

- Recurrent Neural Networks take the previous output or hidden states as inputs
- The composite input at time "t" has some historical information about the happenings at time T < "t".
- RNNs are useful as their intermediate values (state) can store information about past inputs for a time that is not fixed a priori
- Note that the weights are shared over time
- Essentially, copies of the RNN cell are made over time (unrolling/ unfolding), with different inputs at different time steps

That's is Recurrent Neural Network…



We worked out math too….

Recurrent Neural Network

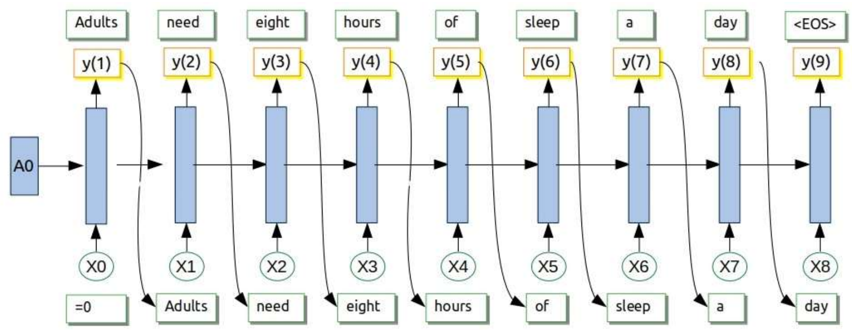
$a_t = g(a_{t-1}.W_{aa} + X_t . W_{ax} + b_a)$

$\hat{y}_t = g(a_t.W_{ya} + b_y)$

5

RNN Model – Sampling from Trained Model

- And the Cost functions
- $J(\hat{y}, y) = \Sigma\ \ell\ (\hat{y},\ y)$ BTW: these cost functions are also known as Jacobians
- $J(\hat{y}, y) = -\frac{1}{m}\Sigma\ y * \log(\hat{y})$
  - Which we will be minimizing



pra-sâmi

6

Gradient - a Difficult Terrain

- Have seen how to compute the gradient descent update for using backprop
- In case of RNN the backprop is through time
- At times, gradient descent completely fails because either they explode or vanish
- It’s hard to learn dependencies over long time windows
- How to learn long-term dependencies?

pra-sâmi

7

"Sentences can be tricky…

"As he crossed toward the pharmacy at the corner he involuntarily turned his head because of a burst of light that had ricocheted from his temple, and saw, with that quick smile with which we greet a rainbow or a rose, a blindingly white parallelogram of sky being unloaded from the van—a dresser with mirrors across which, as across a cinema screen, passed a flawlessly clear reflection of boughs sliding and swaying not arboreally, but with a human vacillation, produced by the nature of those who were carrying this sky, these boughs, this gliding façade."

"

Vladimir Nabokov, "The Gift." 96 words sentence…

5/25/2024

pra-sâmi

8

Vanishing and Exploding Gradients

5/25/2024

pra-sâmi

Vanishing Gradient / Exploding Gradient

9

- What happens to the magnitude of the gradients as we back propagate through many layers?
  - If the weights are small, the gradients shrink exponentially
  - If the weights are big the gradients grow exponentially
- Typical feed-forward neural nets can cope with these exponential effects because they only have a few hidden layers
- We can manage gradients by initializing the weights very carefully in feed-forward networks
  - We have already experienced by using appropriate
- Is it applicable to RNNs as well?

Vanishing Gradient / Exploding Gradient

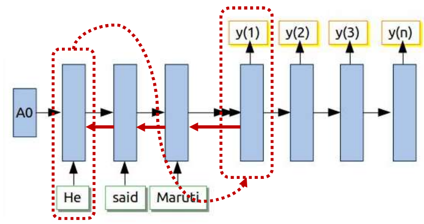
10

- In an RNN trained on long sequences (e.g. 100 time steps) the gradients can easily explode or vanish.
- Even with good initial weights, its very hard to detect that the current target output depends on an input from many time-steps ago
  - So RNNs have difficulty dealing with long-range dependencies

11

Why Gradients Explode or Vanish

- Recall the RNN for machine translation
  - For example, we read an entire English sentence, and then has to output its French translation



- A typical sentence length is 20 words. This means there's a gap of 20 time steps between when we see some information and when we needs it.

- The derivatives need to travel over this entire pathway

pra-sâmi

12

Why Gradients Explode or Vanish…

- Please recall:
  - $z = X . W + b$
  - $\hat{y} = a = \sigma (z)$
  - $a_1 = \sigma (a_0 . W_1)$

- That through the time steps will be
  - $\hat{y} = \sigma (\sigma (\sigma (\sigma (\sigma (a_0 . W_1) . W_2). W_3). W_4)$

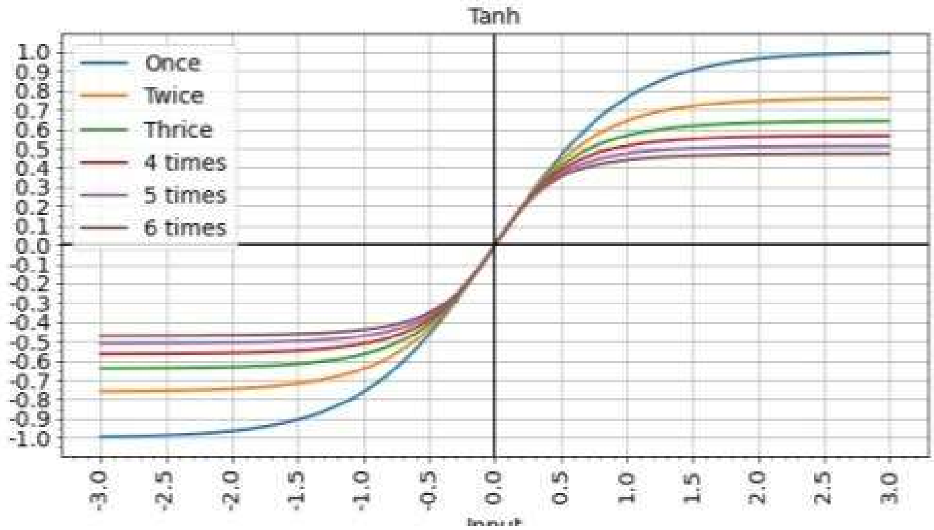
- In backprop, we will be carrying L(ŷ , y) through the activation function iteratively…

- Longer the chain... more iterations on the Ws...

pra-sâmi

13

Iterated functions…

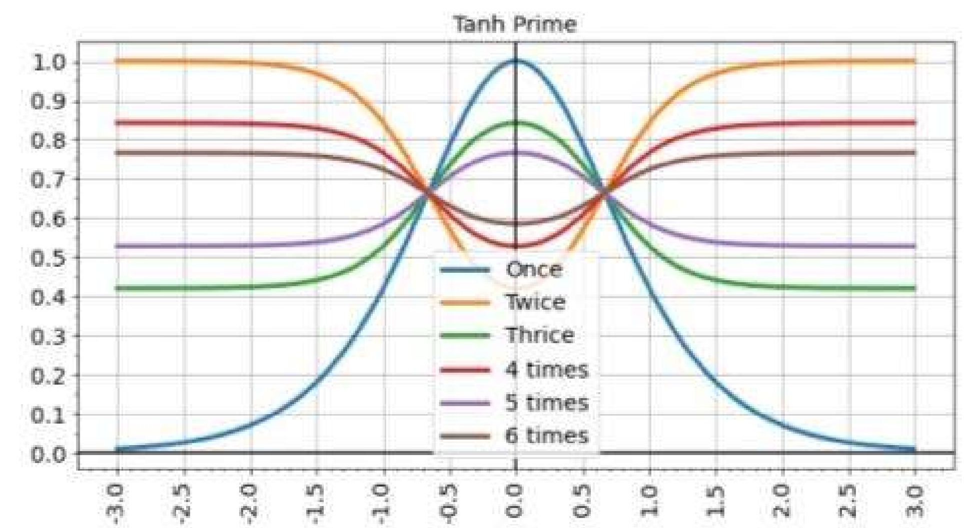


5/25/2024

pra-sâmi

14

Iterated functions…

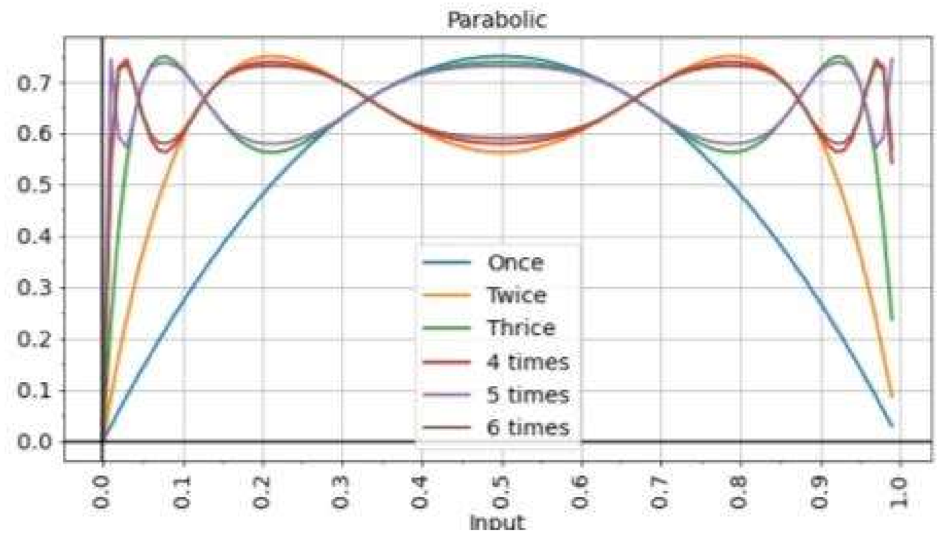


5/25/2024

pra-sâmi

15

Iterated functions…



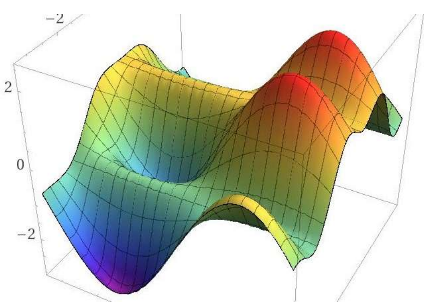
5/25/2024

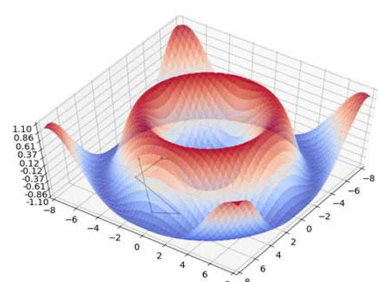
pra-sâmi

16

Why Gradients Explode or Vanish

- Let’s imagine an RNN’s behavior as a dynamical system, which has various slopes and valleys





- Within one of the colored regions, the gradients vanish because even if you move a little, you still wind up at the same valley

- If you’re on the boundary, the gradient blows up because moving slightly moves you from one side to another and subsequently to other valley

5/25/2024

pra-sâmi

Keeping Things Stable

pra-sâmi

Keeping Things Stable - Gradient Clipping

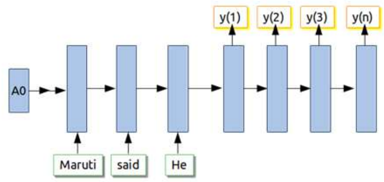
- Clip the gradient
- Clip the gradient ‘g’ so that it has a norm of at most ‘η’:
  - if $||g|| > \eta$: then $g = \eta * \frac{g}{||g||}$
  - Where ‘η’ is another parameter you may want to tune
- The gradients are biased, but at least they don’t blow up.

pra-sâmi

19

Keeping Things Stable - Reverse the Input Sequence

- Applicable in similar languages
  - Hindi → Marathi,
  - English →French,
  - Spanish → Portuguese



- No point in using situation like
  - Hindi → Mandarin or 'Hiragana' or 'Kanji' ; may be 'Katakana'
- This way, there's only one time step between the first word of the input and the first word of the output.
- The network can first learn short-term dependencies between early words in the sentence, and then long-term dependencies between later words.

pra-sâmi

20

Keeping Things Stable – Identity initialization

- Redesign the architecture, since the exploding/vanishing problem highlights a conceptual problem with vanilla RNNs
- The hidden units are a kind of memory. Therefore, their default behavior should be to keep their previous value.
  - The function at each time step should be close to the identity function.
  - It's hard to implement the identity function if the activation function is nonlinear!
- If the function is close to the identity, the gradient computations are stable

pra-sâmi

Keeping Things Stable – Identity initialization

- The identity RNN architecture : [Le et al., 2015. A simple way to initialize recurrent networks of rectified linear units.]
  - The activation functions are all ReLU,
  - Recurrent weights are initialized to the identity matrix
- Proof: This simple initialization trick achieved some neat results;
- For instance, it was able to classify MNIST digits which were fed to the network one pixel at a time, as a length-784 sequence

Applicability

- Discussed three mechanism for training RNNs
  - All pretty widely used.
  - But the identity initialization trick actually eludes to something much more fundamental
  - Keep their previous value, unless it is necessary to change
- Ask: the ability to preserve information over time until it's needed

GRU Units

pra-sâmi

Long Short Term Memory (LSTM)

- Architecture designed to make it easy to remember
- Keep it until it's needed.
- The name refers to the idea:
  - The activations of a network correspond to short-term memory,
    - (Changing very fast with every new incoming record)
  - The weights correspond to long-term memory.
- If the activations can preserve information over multiple time steps
  - That makes them long-term short-term memory
- It's composed of memory cells which have controllers governing when to store or forget information

pra-sâmi

25

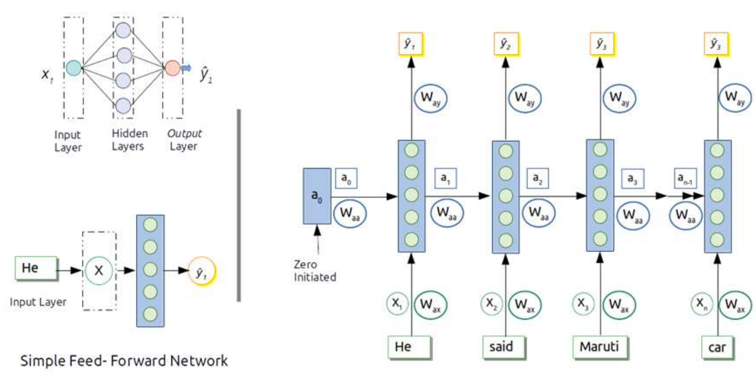
Gated Recurrent Unit

- Before we get to LSTM, lets look at its simplified version…
- Introduced by Cho, et al. in 2014, and Chung et al. in 2014 in their respective papers
- GRU (Gated Recurrent Unit)

pra-sâmi

26

Gated Recurrent Unit

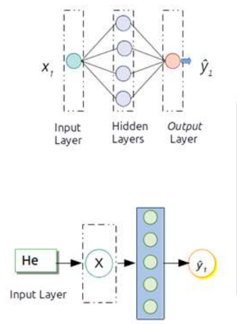


Converted our simple feed forward network to Recurrent Network

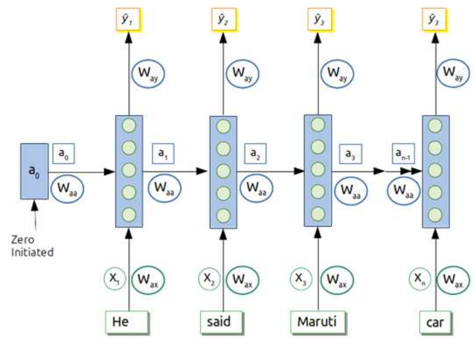
pra-sâmi

Gated Recurrent Unit

27



Simple Feed- Forward Network



and the Math behind it….

Recurrent Neural Network

$a_t = g\,([a_{t-1} : x_t] \,.\, W_a + b_a)$

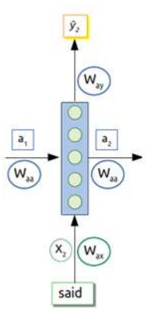
$\hat{y}_t = g\,(a_t \,.\, W_y + b_y)$

5/25/2024

pra-sâmi

Gated Recurrent Unit

28



- RNN Equation
  - $a_t = g_1\,(\,[a_{t-1} : x_t\,] \cdot W_a + b_a)$

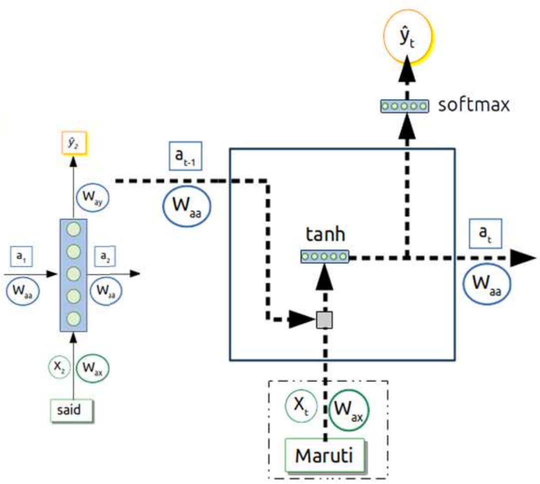
    and
  - $\hat{y}_t = g_2\,(\,a_t \;\cdot W_y + b_y)$
- We look at an alternate way to paint the network

5/25/2024

pra-sâmi

29

RNN



Basic RNN

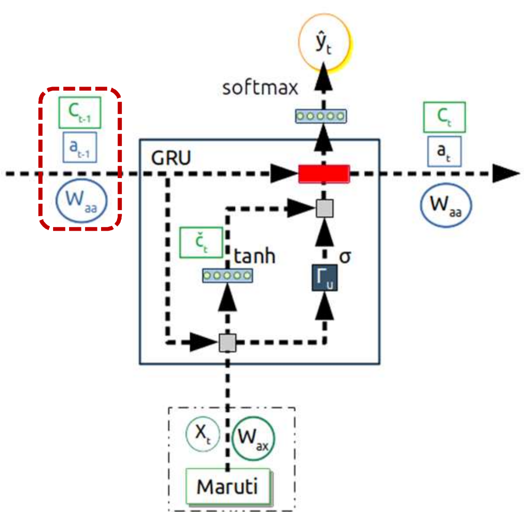
- RNN Equation
  - $a_t = g_1\ ([a_{t-1} : x_t\ ] \cdot W_a + b_a)$
    and
  - $\hat{y}_t = g_2\ (\ a_t\ \cdot W_y + b_y)$
- Alternate layout for one of the recurrence

5/25/2024

pra-sâmi

30

GRU Cell



Gated recurrent units (GRUs)

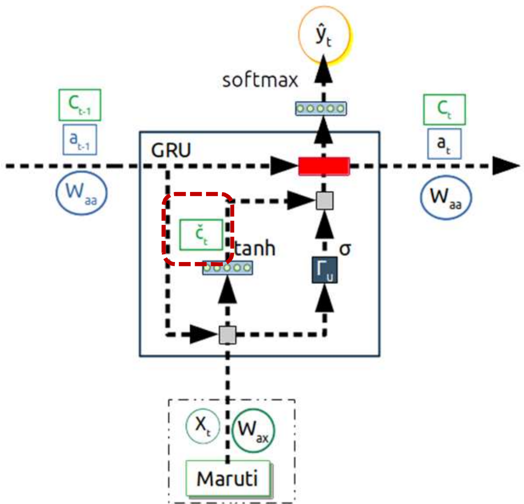
- GRU will have a cell which we will use to **retain** values from earlier iterations.
- Cell is called memory cell and is represented by ‘c’.
- In this case , memory cell value ‘c’ and activation ‘a’ value will be same.

5/25/2024

pra-sâmi

GRU Cell

31



Gated recurrent units (GRUs)

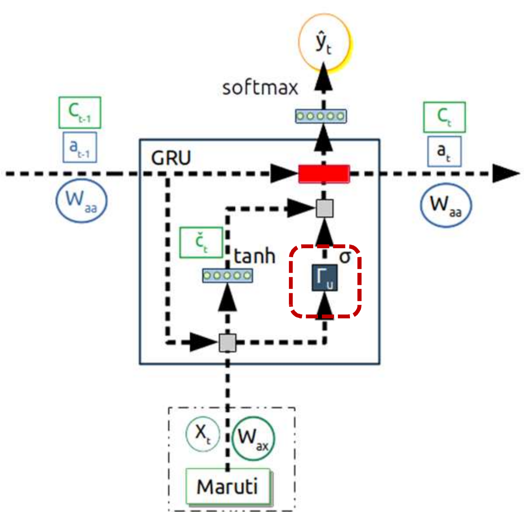
- At every time step, we will calculate 'ĉ', a candidate which may replace 'c_{t-1}'
  - $\hat{c}_t$ = tanh ($[c_{t-1} : x_t]$. W_c+ b_c)
- May be not…
- Next we need some parameter which will tell us if we need to replace the value of 'c_t' with candidate '$\hat{c}_t$' or not
- Note: At present both $c_t$and a_t are same… keeping them separate for consistency

5/25/2024

pra-sâmi

GRU Cell

32



Gated recurrent units (GRUs)

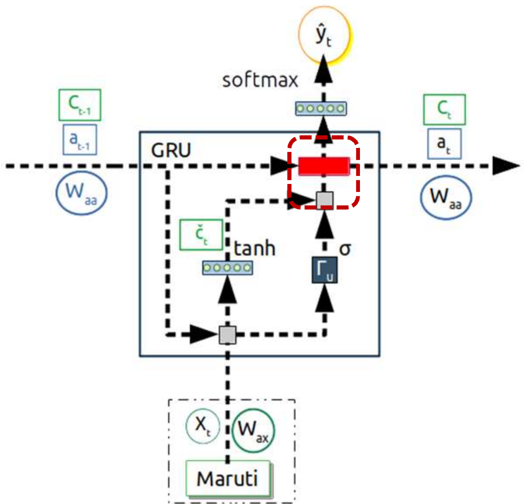
- Important change is called ' Gate'
- Gate, represented by 'Γ' Gamma is:
  - Γ_u = σ ([c_{t-1} : x_t] . W_u+ b_u)
- In above equation, we are using separate Weights and Biases for update gate.
- The activation used here is sigmoid so for most part our Gamma will be 1 or 0

5/25/2024

pra-sâmi

33

GRU Cell



Gated recurrent units (GRUs)

- With ‘Γ_u’ Gamma update, we can calculate c_t as follows:
- $c_t = \Gamma_u * ĉ_t$ + (1-Γ_u) * c_{t-1}
- Intuition: how Γ_u will be used????

“I **felt happy** because I saw the others **were happy** and because I knew I should **feel happy,** but I **wasn’t really happy.”**

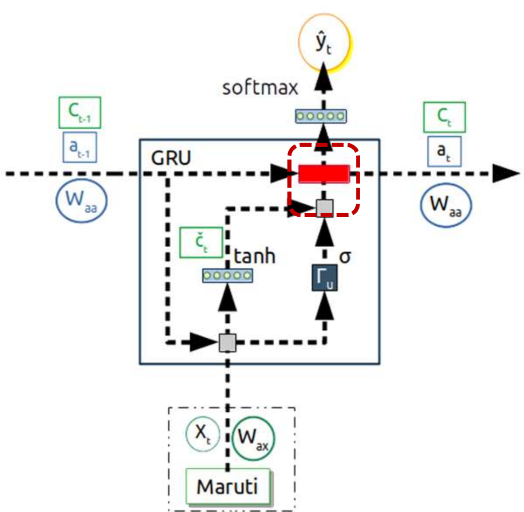
- Create or keep c_t
- Time to Replace c_t

5/25/2024

pra-sâmi

34

GRU Cell



Gated recurrent units (GRUs)

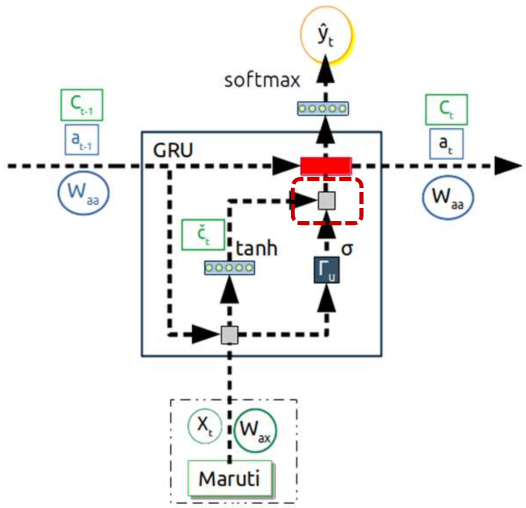
- So we can maintain c_t and their values through many layers and use them repeatedly.
- And more importantly, when to change it
- I **felt happy** because I saw the others **were happy** and because I knew I should **feel happy,** but I **wasn’t really happy.**

5/25/2024

pra-sâmi

35

GRU Cell

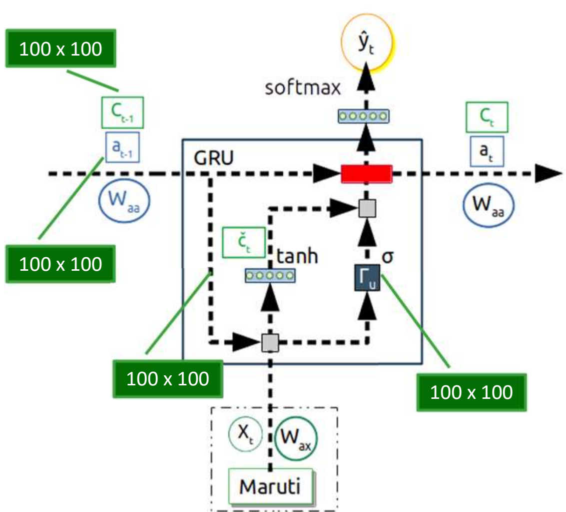


Gated recurrent units (GRUs)

- That was GRU in simplified form.
- Actual implementation has one additional parameter Γ_r Gamma Relevance
  - $\hat{c}_t = \tanh([\Gamma_r * c_{t-1} : x_t] . W_c + b_c)$
  - $\Gamma_u = \sigma([c_{t-1} : x_t] . W_u + b_u)$
  - $\Gamma_r = \sigma([c_{t-1} : x_t] . W_r + b_r)$
  - $c_t = \Gamma_u * \hat{c}_t + (1 - \Gamma_u) * c_{t-1}$

36

GRU Cell



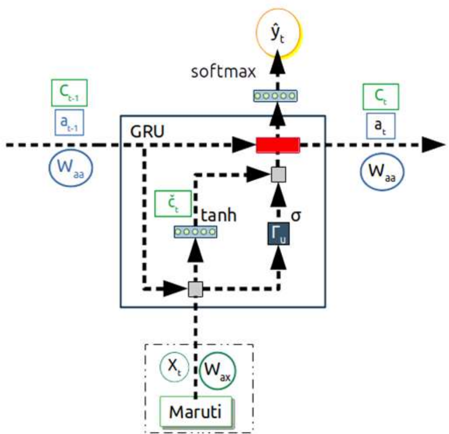
Gated recurrent units (GRUs)

- That was GRU in simplified form.
- Actual implementation has one additional parameter Γ_r Gamma Relevance
  - $\hat{c}_t = \tanh([\Gamma_r * c_{t-1} : x_t] . W_c + b_c)$
  - $\Gamma_u = \sigma([c_{t-1} : x_t] . W_u + b_u)$
  - $\Gamma_r = \sigma([c_{t-1} : x_t] . W_r + b_r)$
  - $c_t = \Gamma_u * \hat{c}_t + (1 - \Gamma_u) * c_{t\ 1}$

Element wise multiplications

37

GRU Cell



Gated recurrent units (GRUs)

Extended GRU:

$\check{c}_t = \tanh([\Gamma_r * c_{t-1} : x_t] . W_c + b_c)$

$\Gamma_u = \sigma([c_{t-1} : x_t] . W_u + b_u)$

$\Gamma_r = \sigma([c_{t-1} : x_t] . W_r + b_r)$

$c_t = \Gamma_u . \check{c}_t + (1 - \Gamma_u) . c_{t-1}$

If $\Gamma_u = 1$ then c_t will be equal to $\check{c}_t$,

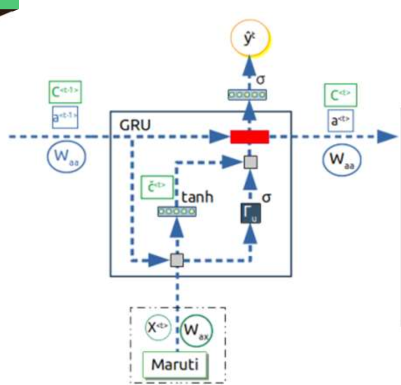
If $\Gamma_u = 0$ then c_t will be equal to c_{t-1}

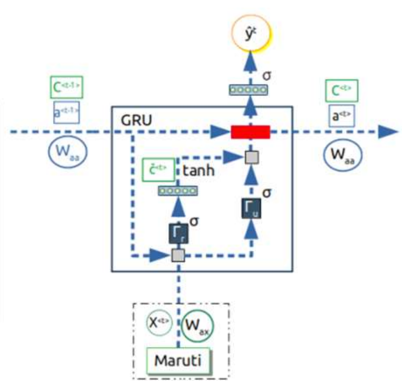
5/25/2024

pra-sâmi

38

GRU Cell





$a_t = g^1(a_{t-1} . W_{aa} + X_t . W_{ax} . + b_a)$

[100,100] [100, 100] [100,10000] [10000, 100]

$\hat{y}_t = g^2(a_t . W_{ya} . + b_y)$

$a_t = g^1([a_{t-1} : X_t] . W_a . + b_a)$

$\hat{y}_t = g^2(a_t . W_y + b_y)$

$[a_{t-1}, X_t] = [a_{t-1} : X_t]$ [100]

[100 x 10100] [100] [10000]

$$W_a = \begin{array}{c} [100] \\ \begin{pmatrix} W_{aa} \\ W_{ax} \end{pmatrix} \end{array} \begin{array}{l} [100] \\ [10000] \end{array}$$

Simple GRU:

$\check{C}^{<t>} = \tanh([C^{<t-1>} : X^{<t>}] . W_c + b_c)$

$\Gamma_u = \sigma([C^{<t-1>} : X^{<t>}] . W_u + b_u)$

$C^{<t>} = \Gamma_u . \check{C}^{<t-1>} + (1 - \Gamma_u) . C^{<t-1>}$

If $\Gamma_u = 1$ then $C^{<t>}$ will be equal to $\check{C}^{<t>}$,

If $\Gamma_u = 0$ then $C^{<t>}$ will be equal to $C^{<t-1>}$.

GRU:

$\check{C}^{<t>} = \tanh([\Gamma_r * C^{<t-1>} : X^{<t>}] . W_c + b_c)$

$\Gamma_u = \sigma([C^{<t-1>} : X^{<t>}] . W_u + b_u)$

$\Gamma_r = \sigma([C^{<t-1>} : X^{<t>}] . W_r + b_r)$

$C^{<t>} = \Gamma_u . \check{C}^{<t>} + (1 - \Gamma_u) . C^{<t-1>}$

If $\Gamma_r = 1$ then consider $C^{<t-1>}$ in $\check{C}^{<t>}$ calculations,

If $\Gamma_r = 0$ then do not consider $C^{<t-1>}$ in $\check{C}^{<t>}$ calculations.

If $\Gamma_u = 1$ then $C^{<t>}$ will be equal to $\check{C}^{<t>}$,

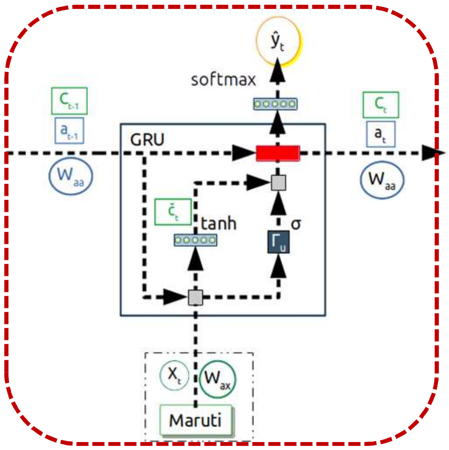
If $\Gamma_u = 0$ then $C^{<t>}$ will be equal to $C^{<t-1>}$.

5/25

pra-sâmi

39

GRU Cell



Gated recurrent units (GRUs)

- And that, my friends, is GRU….

  $\hat{c}_t = \tanh([\Gamma_r * c_{t-1} : x_t] . W_c + b_c)$

  $\Gamma_u = \sigma([c_{t-1} : x_t] . W_u + b_u)$

  $\Gamma_r = \sigma([c_{t-1} : x_t] . W_r + b_r)$

  $c_t = \Gamma_u * \hat{c}_t + (1 - \Gamma_u) * c_{t-1}$

- Coming up next… LSTM! After a notebook….

5/25/2024

pra-sâmi

40

5/25/2024

pra-sâmi