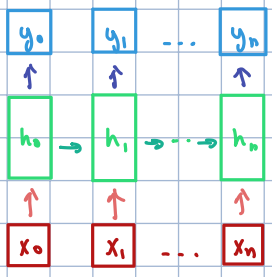
RNN: why

- vanilla neural nets (& CNNs) take fixed len in, produce fixed len out
- RNN can do variable len

How work? (many to many example)

- input: $x_0, x_1, \dots, x_n$
- output: $y_0, y_1, \dots, y_n$ ← x_i, y_i are vectors of arbitrary dim.
- steps: iteratively update hidden state h (matrix w/ arbitrary dim). At step t:
  - 1) next hidden state calculated using h_{t-1} & next input x_t



each h_n takes h_{n-1}, x_n as input

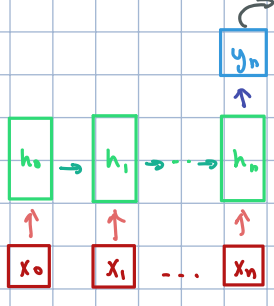
- vanilla RNN: 3 sets of weights
  - W_{xh}: $x_t \rightarrow h_t$ links
  - W_{hh}: $h_{t-1} \rightarrow h_t$
  - W_{hy}: $h_t \rightarrow y_t$
  - biases: b_h (for calc h_t)
    b_y (for calc y_t)
  - representations: weights as MATRICES, biases as VECTORS
  - calculation:
    - $h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$ ← can use another activation function like sigmoid
    - $y_t = W_{hy}h_t + b_y$

The problem

- sentiment analysis: determine if text is pos. or neg.
- ex: I am good ⇒ good
  I am not happy ⇒ bad

Plan

- use "many to one" RNN
- final output will just be the last y_n
- format:



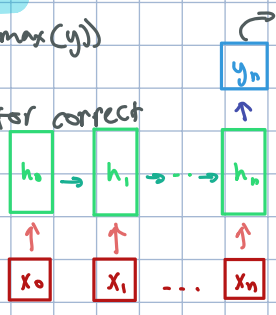
← vector w/ 2 values: 1 pos, 1 neg.
use SOFTMAX to convert → probability

← x_i: 1 word from the text

sizes: x_i: input_size x 1 ← input_size = vocab_size
W_{xh}: hidden size x input_size b_h: hidden_size x 1
W_{hh}: hidden size x hidden size W_{hy}: output_size x hidden size
h_t, h_{t-1}: hidden size x 1 b_y: output size x 1

Backprop

- can use cross-entropy loss, which is often paired w/ softmax
  $L = -\ln(p_c)$
  p_c = RNN's predicted probability for the correct class
  ex: if pos. text is predicted as 90% positive by RNN, loss is:
  $L = -\ln(.90) = .105$
  using cross-entropy guides the RNN to 1) get the correct ans & 2) be more confident in its answer
  ex: if pos text is predicted as 10% positive by RNN:
  1) it's wrong
  2) it's not confident.
  ∴ incurs large loss of 2.3026
- defn:
  y = RAW outputs from RNN
  p = probabilities ($p = \text{softmax}(y)$)
  c = correct label
  L = cross entropy loss ($L = -\ln(p_c)$)
  W_{xh}, W_{hh}, W_{hy} = 3 weight matrices
  b_h, b_y = 2 bias vectors
- gradients:
  start by calc dL/dy:
  $L = -\ln(p_c) = -\ln(\text{softmax}(y))$



p_c = RNN's predicted prob for correct class.

← vector w/ 2 values: 1 pos, 1 neg. use softmax to convert → probability

← x_i : 1 word from the text

y = [probability of positive, prob. of neg]

ex: [.90, .10] means 90% chance positive

↑ p = probability of positive

$L = c(-\ln(p)) + (1-c)(-\ln(1-p))$

↳ if c (correct label) is 1 (aka positive) return $(-\ln(p))$. the $(1-c)(-\ln(1-p))$ part eval to 0
if c is 0, return $-\ln(1-p)$, aka $-\ln(\text{prob\_neg})$
the $c(-\ln(p))$ term evaluates to 0

y_p = RNN's predicted chance of phrase being positive

$$\frac{dL}{dy_p} = \frac{d}{dy_p}\left[c(-\ln p) + (1-c)(-\ln(1-p))\right]$$

$$= -c\frac{d}{dy_p}\ln(\text{softmax}(y_p)) - (1-c)\frac{d}{dy_p}(\ln(1-p))$$

$= \frac{\partial L}{\partial y_p}$

$$\frac{d}{dy_p} \ln(sm(y_p)) = \frac{1}{sm(y_p)} \cdot \frac{d}{dy_p} sm(y_p)$$

$$= \frac{\sum_{i=1}^{n} e^{y_i}}{e^{y_p}} \cdot \frac{d}{dy_p}\left(\frac{e^{y_p}}{\sum_{j=1}^{n} e^{y_i}}\right) \quad \text{quotient rule}$$

$$= \frac{\sum_{i=1}^{n} e^{y_i}}{e^{y_p}} \cdot \frac{e^{y_p}\left(\sum e^{y_i}\right) - (e^{y_p})^2}{\left(\sum e^{y_i}\right)^2}$$

$$= 1 - sm(y_p)$$

$$\frac{dL}{dy_p} = c(sm(y_p) - 1) - (1-c)\frac{1}{1-sm(y_p)} \cdot \frac{d}{dy_p}(1 - sm(y_p))$$

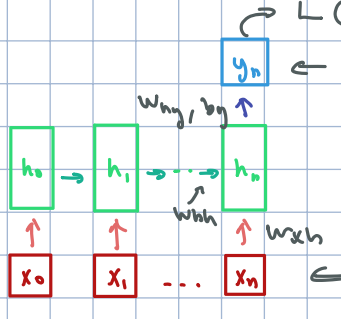
$$\frac{dL}{dy_p} = c(sm(y_p) - 1) - \frac{(1-c)}{1-sm(y_p)}\left[\frac{-e^{y_p}\left(\sum e^{y_i} - e^{y_p}\right)}{\left(\sum e^{y_i}\right)^2}\right]$$

$$+ \frac{(1-c)}{\frac{\sum e^{y_i} - e^{y_p}}{\sum e^{y_i}}}\left[\frac{+e^{y_p}\left(\sum e^{y_i} - e^{y_p}\right)}{\left(\sum e^{y_i}\right)^2}\right]$$

$$= c(sm(y_p) - 1) + (1-c)(sm(y_p))$$

or in other words:

$$\frac{\partial L}{\partial y_i} = \begin{cases} p_i - 1 & \text{if } i = c \\ p_i & \text{if } i \neq c \end{cases}$$



← vector w/ 2 values: 1 pos, 1 neg.
use softmax to convert → probability

← x_i : 1 word from the text

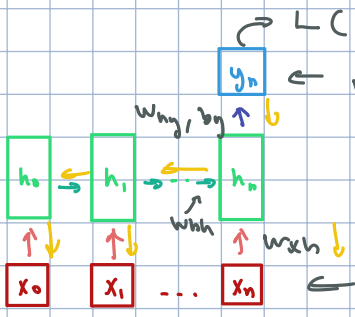
compute gradient for w_{hy}, b_y:

$$\frac{\partial L}{\partial w_{hy}} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial w_{hy}} \qquad \frac{\partial L}{\partial b_y} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial b_y}$$

$y = w_{hy} h_n + b_y$ $\qquad$ $y = w_{hy} h_n + b_y$

$\frac{\partial y}{\partial w_{hy}} = h_n$ $\qquad$ $\frac{\partial y}{\partial b_y} = 1$

$\therefore \frac{\partial L}{\partial w_{hy}} = \frac{\partial L}{\partial y} \cdot h_n$ $\qquad$ $\therefore \frac{\partial L}{\partial b_y} = \frac{\partial L}{\partial y}$

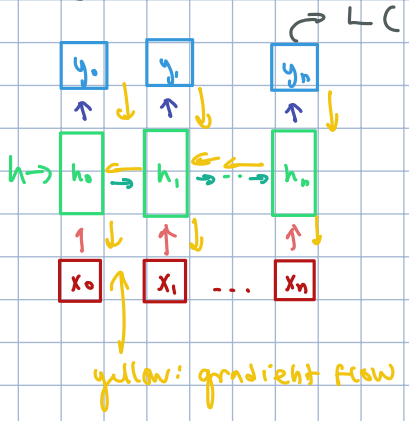


← vector w/ 2 lines: 1 pos, 1 neg.
use softmax to convert → probability

← x_i : 1 word from the text

how do gradient for the weight?

full diagram (if we calculate each y_n) would be:



so aggregate loss (FL) becomes

$$FL = \sum L(y_t)$$

$$\frac{\partial FL}{\partial W_{xh}} = \sum \frac{\partial L(y_t)}{\partial y_t} \cdot \frac{\partial y_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial W_{xh}}$$

as approximation, just have all the y's equal the final y

$$\Rightarrow \frac{\partial FL}{\partial W_{xh}} \approx \frac{\partial L}{\partial W_{xh}} = \frac{\partial L}{\partial y} \sum \frac{\partial y}{\partial h_t} \cdot \frac{\partial h_t}{\partial W_{xh}}$$

calculate $\frac{\partial h_t}{\partial W_{xh}}$:

$h_t = \tanh(W_{xh} x_t + W_{hh} h_{t-1} + b_h)$

$\frac{d \tanh(x)}{dx} = 1 - \tanh^2(x)$

$\therefore$

$$\frac{\partial h_t}{\partial W_{xh}} = (1 - h_t^2)\, x_t$$

calculate $\frac{\partial h_t}{\partial W_{hh}}$:

$$\frac{\partial h_t}{\partial W_{hh}} = (1 - h_t^2)\, h_{t-1}$$

chain rule

calculate $\frac{\partial h_t}{\partial b_h}$:

$$\frac{\partial h_t}{\partial b_h} = (1 - h_t^2)$$

$$\frac{\partial L}{\partial W_{hh}} = \frac{\partial L}{\partial y} \sum \frac{\partial y}{\partial h_t} \cdot \frac{\partial h_t}{\partial W_{hh}}$$

&

$$\frac{\partial L}{\partial b_h} = \frac{\partial L}{\partial y} \sum \frac{\partial y}{\partial h_t} \cdot \frac{\partial h_t}{\partial b_h}$$

calculate $\frac{\partial y}{\partial h_t}$

$$\frac{\partial y}{\partial h_t} = \frac{\partial y}{\partial h_{t+1}} \cdot \frac{\partial h_{t+1}}{\partial h_t}$$

ex: $\frac{\partial y}{\partial h_0} = \frac{\partial y}{\partial h_1} \cdot \frac{\partial h_1}{\partial h_0}$

$$\frac{\partial y}{\partial h_t} = \frac{\partial y}{\partial h_{t+1}} \cdot \frac{\partial h_{t+1}}{\partial h_t} = \frac{\partial y}{\partial h_{t+1}} \cdot (1 - h_{t+1}^2) W_{hh}$$

but $\frac{\partial y}{\partial h_n} = W_{hy}$ ← only true for final multiplication

finish: $\frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial h} = \frac{\partial L}{\partial h}$

$$\frac{\partial L}{\partial w_{xh}} = \sum \frac{\partial L}{\partial h} (1-h_t^2) x_t$$

$$\frac{\partial L}{\partial w_{hh}} = \sum \frac{\partial L}{\partial h} (1-h_t^2) h_{t-1}$$

$$\frac{\partial L}{\partial b_h} = \sum \frac{\partial L}{\partial h} (1-h_t^2)$$

$$\frac{\partial L}{\partial h} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial h}$$

$$temp = (1-h_{t+1}^2) \frac{\partial L}{\partial h}$$

$$next \quad \frac{\partial y}{\partial h} : \frac{\partial y}{\partial h} (1-h_t^2) w_{hh}$$

$$next \quad \frac{\partial L}{\partial h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial h} (1-h_t^2) w_{hh}$$

$$= \frac{\partial L}{\partial h} (1-h_t^2) w_{hh}$$

$$= (temp) w_{hh}$$