CS 378 Lecture 16

Announcements

Assignments: A4 only coding (but simplified), A5 no coding,
FP in progress (w/partner possible, no pres.)

Extra slip days

Zoom protocol: type Qs in chat, polls on Canvas, breakouts

Recap

Language modeling: $P(\bar{w})$ prob of sentence or document

n-gram model: $P(\bar{w}) = \prod_{i=1}^{n} P(w_i \mid w_{i-n+1}, \ldots, w_{i-1})$

3-gram prob of "I want to go":

$P(I \mid \text{<s> <s>})\, P(\text{want} \mid \text{<s> } I)\, P(\text{to} \mid I \text{ want})$
$P(\text{go} \mid \text{want to})\, P(\text{STOP} \mid \text{to go})$

Estimate counts from large corpora, smooth

Goals Recurrent neural networks (RNN)

1. key RNN abstraction
2. Key properties of LSTM (long short-term memory)

This lecture: LM, RNN definition

Next lecture: training + implementation

This sets the stage for machine translation, summarization, dialogue

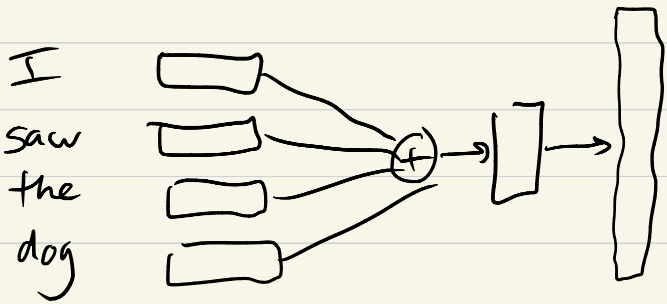
Neural Language Models

$\in \overset{\text{vocab}}{V}$

Discrim model for $P(w_i \mid \underbrace{w_1 \ldots w_{i-1}}_{\text{all past words}})$

What we want: neural net to look at $w_1 \ldots w_{i-1}$, place distribution over w_i

DAN: + fast — ignores order



$P(w_i \mid w_1 \ldots w_{i-1})$

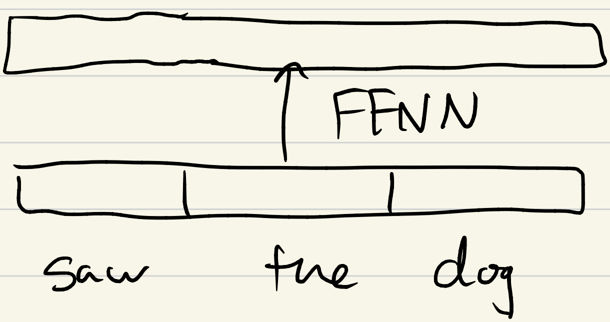
running
barking

Feedforward n-gram model

4-gram model:

+ uses order
- not much context

$P(w_i \mid w_{i-3}, w_{i-2}, w_{i-1})$



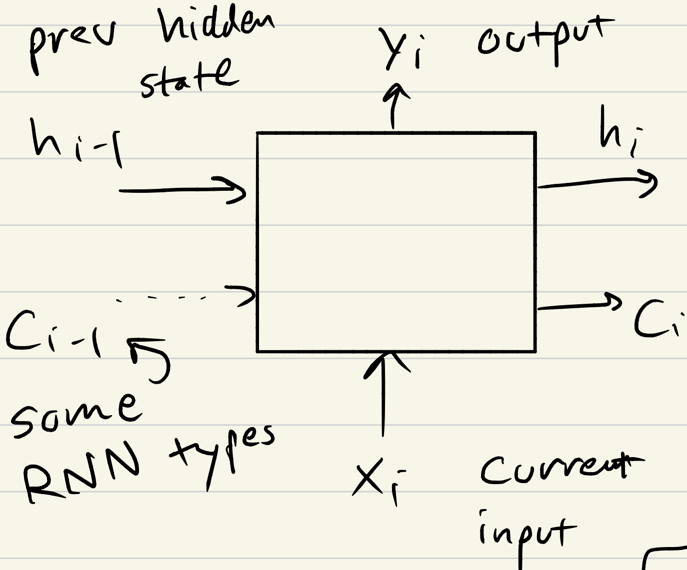
past 3 words

input: |word emb| · (n-1)
need large hidden states (1000+)

3 ⇒ 10: more params in NN

params: |word emb| · (n-1) · 1000 + ...

<u>RNN</u>: neural model for encoding seqs of arbitrary length



x, y, h vectors

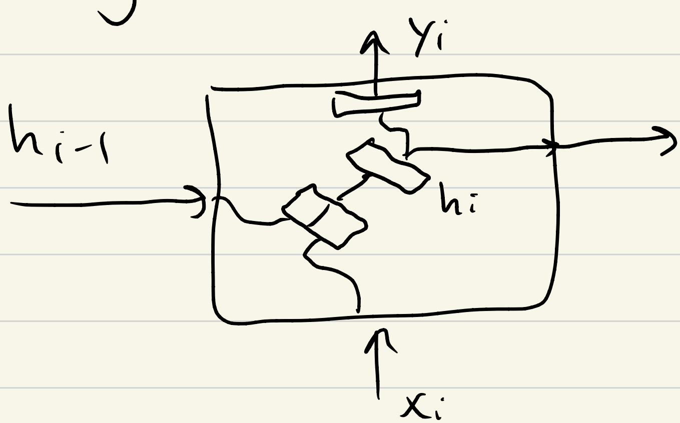
h_{i-1}, x_i inputs

y_i, h_i outputs

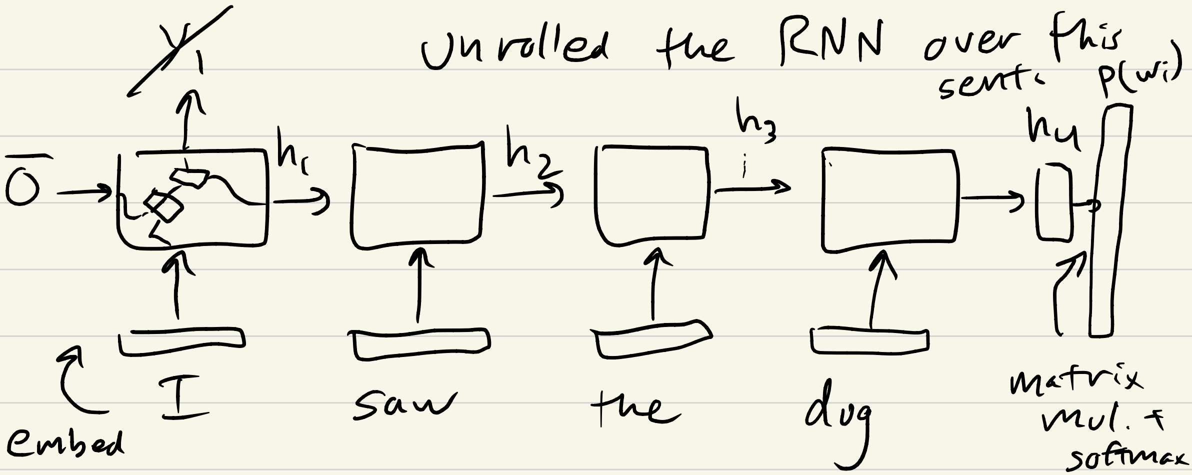
$$\boxed{\begin{aligned} h_i &= \tanh(Wx_i + Vh_{i-1}) \\ y_i &= \tanh(Uh_i) \end{aligned}}$$

Elman network
(UT ling PhD 1977)

W, V, U
parameter matrices



Unrolled the RNN over this sent.



forward() in PyTorch has a for loop, but otherwise this is just a special feedforward network

At each step, hi captures the model's "picture" of the sentence so far

$$P(w_i \mid w_1 \ldots w_{i-1}) = \text{softmax}(M h_{i-1})$$

$$h_{i-1} = \text{RNN}(w_1 \ldots w_{i-1})$$

$|V| \cdot |h|$

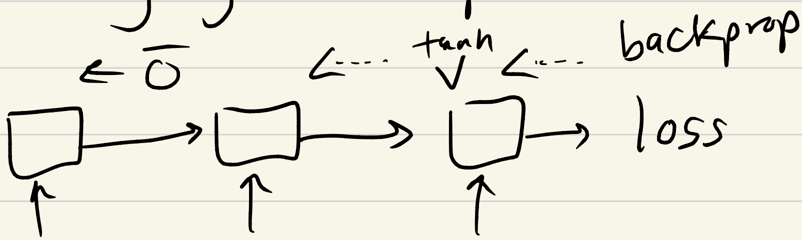
W, U, V, M, are our only params!

Training RNNs

"Backpropagation through time" = backpropagation
More next time

Long short-term memory network

Vanishing gradient problem



Elman networks (not LSTMs): error term "dies out"

model can't learn to feed info long distance

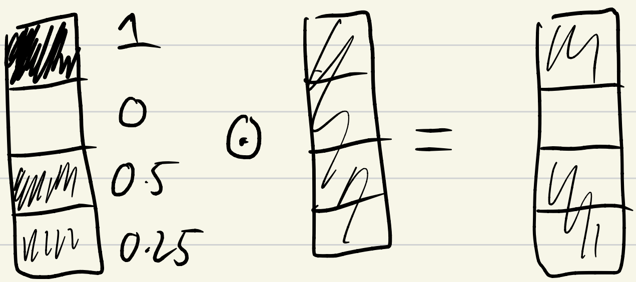
Key idea: gates

Elman: $h_i = \tanh(Wx_i + Vh_{i-1})$

Gated: $h_i = h_{i-1} \odot f + \text{func}(x_i) \odot i$

$\odot$ = element-wise mul

f: vector $\in [0,1]^d$



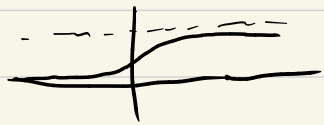
f $\quad\quad$ $h_{i-1} \in \mathbb{R}^d$

$f = I$ preserves h_i: $h_i = \sum_{j=0}^{i-1} \text{func}(x_i) \odot i$

otherwise zeroes out some parts

$$\bar{f} = \underset{\text{elementwise}}{\text{sigmoid}}\left(W^1 \bar{x}_i + W^2 \bar{h}_{i-1}\right) \quad \text{+bias}$$

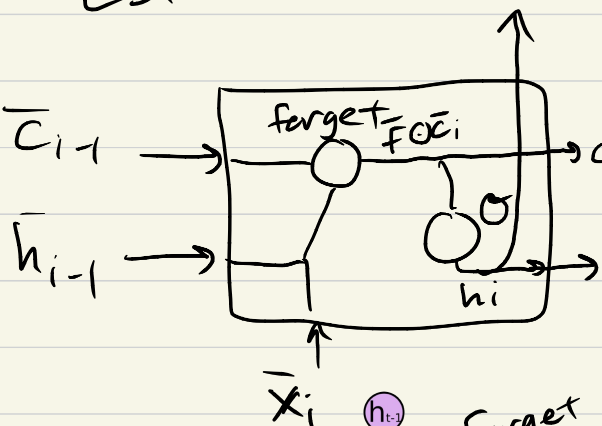
$$\sigma = \frac{e^x}{1+e^x}$$



$$\bar{i} = \text{sigmoid}\left(W^3 \bar{x}_i + W^4 \bar{h}_{i-1}\right)$$

LSTM:

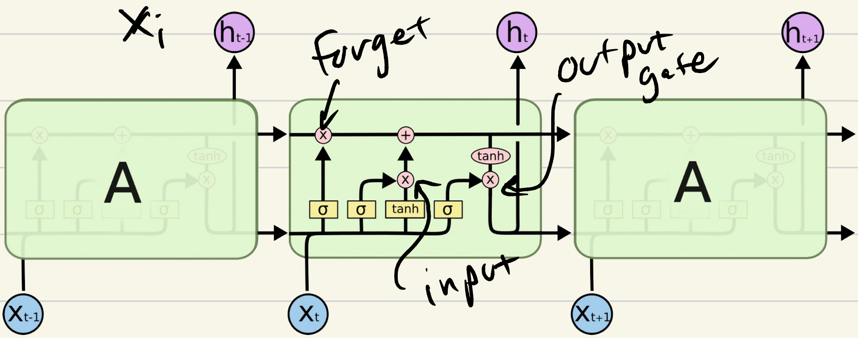
$\bar{c}$: short-term memory "cell"



$$\bar{c}_i = \bar{c}_{i-1} \odot \bar{f} + \text{func}(x_i) \odot \bar{i}$$

$$\bar{h}_i = \bar{c}_i \odot \bar{o}$$

another gate



Source: Chris Olah blog post (link on course website)

Key properties:

RNN: seq of words (vectors) as input

encodes into State h_i

LSTM: 2 hidden states (h, c)

h vs. c distinction is not critical

better at remembering info for long sequences by using gates rather than tanh + matrix mul.

Resources: Chris Olah blog

PyTorch example lstm_lecture.py

Floydhub tutorial