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: \( P(\mathbf{w}) \) prob of sentence or document

\[ n\text{-gram model: } P(\mathbf{w}) = \prod_{i=1}^{n} P(w_i | w_{i-n+1}, \ldots, w_{i-1}) \]

3-gram prob of “I want to go”:
\[
P(\text{I I less}) P(\text{want I less I}) P(\text{to I want}) P(\text{go I want to}) P(\text{stop I to go})
\]

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 and implementation
This sets the stage for machine translation, summarization, dialogue

Neural Language Models

Discrim model for $P(w_i | w_1, ..., w_{i-1})$

What we want: neural net to look at $w_1, ..., w_{i-1}$, place distribution over $w_i$
DAN: + Fast  - ignores order

I saw the dog

Feedforward n-gram model  + uses order
U-gram model:

\[ P(w_i | w_{i-3}, i-2, i-1) \]

input: (word emb) \cdot (n-1)  \Rightarrow 10: more
need large hidden states, params in NN (1000+)

params: | word emb| (n-1) \cdot 1000 + \ldots
RNN: neural model for encoding sets of arbitrary length

\[ h_i = \tanh(Wx_i + Vh_{i-1}) \]
\[ y_i = \tanh(Uh_i) \]

Elman network (UT Ling PhD 1977)
At each step, $h_i$ captures the model’s “picture” of the sentence so far.

$$P(w_i | w_1, ..., w_{i-1}) = \text{softmax} \left( M h_{i-1} \right)$$

$$h_{i-1} = \text{RNN}(w_i, ..., w_{i-1})$$

$w, U, V, M$ are our only parameters!
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: \( h_i = h_{i-1} \odot f + \text{func}(x_i) \odot i \)

\( \odot = \text{element-wise mul} \)

\( f: \text{vector } \mathbb{E} [0, 1]^d \)

\[
\begin{pmatrix}
1 & 0 & 0.5 & 0.25
\end{pmatrix}
= \begin{pmatrix}
1 & 0 & 0 & 0
\end{pmatrix}
\]

\( f = I \) preserves \( h_i: h_i = \sum_{j=0}^{\infty} \text{func}(x_i) i \)

otherwise zeroes out some parts
- element-wise

\[ F = \text{sigmoid} \left( W' \bar{x}_i + W^2 \bar{h}_{i-1} \right) \]

\[ o = \frac{e^x}{1+e^x} \]

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

LSTM:

\[ \bar{c}_i = \bar{c}_{i-1} \circ F + \text{func}(x_i) \]

\[ h_i = c_i \circ o \]

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_example)
- Floydhub tutorial