From Recurrent Language Models to Machine Translation

CMSC 470
Marine Carpuat
A recurrent language model

\[ m_t = M \cdot e_{t-1} \]

\[ h_t = \begin{cases} \tanh(W_{mh}m_t + W_{hh}h_{t-1} + b_h) & t \geq 1, \\ 0 & \text{otherwise}. \end{cases} \]

\[ p_t = \text{softmax}(W_{hs}h_t + b_s). \]
A recurrent language model

\[ m_t = M_{:,t-1} \]
\[ h_t = \text{RNN}(m_t, h_{t-1}) \]
\[ p_t = \text{softmax}(W_{hs} h_t + b_s). \]
Examples of RNN variants

- **LSTMs**
  - Aim to address vanishing/exploding gradient issue

- **Stacked RNNs**

- ...
Training in practice: online

**Algorithm 1** A fully online training algorithm

1: **procedure** ONLINE
2:     for several epochs of training **do**
3:         for each training example in the data **do**
4:             Calculate gradients of the loss
5:         Update the parameters according to this gradient
6:     **end for**
7: **end for**
8: **end procedure**
Training in practice: batch

Algorithm 2 A batch learning algorithm

1: procedure BATCH  
2: \hspace{1em} for several epochs of training do  
3: \hspace{2em} for each training example in the data do  
4: \hspace{3em} Calculate and accumulate gradients of the loss  
5: \hspace{2em} end for  
6: \hspace{1em} Update the parameters according to the accumulated gradient  
7: \hspace{1em} end for  
8: end procedure
Training in practice: minibatch

• Compromise between online and batch

• Computational advantages
  • Can leverage vector processing instructions in modern hardware
  • By processing multiple examples simultaneously
Problem with minibatches: in language modeling, examples don’t have the same length

- **3 tricks**
  - **Padding**
    - Add </s> symbol to make all sentences same length
  - **Masking**
    - Multiply loss function calculated over padded symbols by zero
  - + sort sentences by length
Machine Translation

- Translation system
  - Input: source sentence $F$
  - Output: target sentence $E$
  - Can be viewed as a function

\[ \hat{E} = \text{mt}(F) \]

- Statistical machine translation systems

\[ \hat{E} = \underset{E}{\text{argmax}} \ P(E \mid F; \theta) \]

- 3 problems

  - Modeling
    - how to define $P(.)$?

  - Training/Learning
    - how to estimate parameters from parallel corpora?

  - Search
    - How to solve argmax efficiently?
Encoder-decoder model
Encoder-decoder model

\[ m^{(f)}_t = M^{(f)}_{t, f_t} \]

\[ h^{(f)}_t = \begin{cases} \text{RNN}^{(f)}(m^{(f)}_t, h^{(f)}_{t-1}) & t \geq 1, \\ 0 & \text{otherwise.} \end{cases} \]

\[ m^{(e)}_t = M^{(e)}_{t, c_{t-1}} \]

\[ h^{(e)}_t = \begin{cases} \text{RNN}^{(e)}(m^{(e)}_t, h^{(e)}_{t-1}) & t \geq 1, \\ h^{(f)}_{|F|} & \text{otherwise.} \end{cases} \]

\[ p^{(e)}_t = \text{softmax}(W_{hs} h^{(e)}_t + b_s) \]
Generating Output

• We have a model $P(E|F)$, how can we generate translations?

• 2 methods

  • **Sampling**: generate a random sentence according to probability distribution

  • **Argmax**: generate sentence with highest probability
Ancestral Sampling

• Randomly generate words one by one

• Until end of sentence symbol

• Done!

\[
\text{while } y_{j-1} \neq "<\text{/s}>": \\
y_j \sim P(y_j | X, y_1, ..., y_{j-1})
\]
Greedy search

• One by one, pick single highest probability word

while $y_{j-1} \neq \"</s>\":$

\[ y_j = \text{argmax } P(y_j | X, y_1, \ldots, y_{j-1}) \]

• Problems
  • Often generates easy words first
  • Often prefers multiple common words to rare words
Greedy Search

Example
Beam Search

Example with beam size $b = 2$

We consider $b$ top hypotheses at each time step
Other encoder structures: Bidirectional encoder

\[ \overrightarrow{h}_t^{(f)} = \begin{cases} \overrightarrow{\text{RNN}}^{(f)}(m_t^{(f)}, \overrightarrow{h}_{t-1}^{(f)}) & t \geq 1, \\ 0 & \text{otherwise}. \end{cases} \]

\[ \overleftarrow{h}_t^{(f)} = \begin{cases} \overleftarrow{\text{RNN}}^{(f)}(m_t^{(f)}, \overleftarrow{h}_{t+1}^{(f)}) & t \leq |F|, \\ 0 & \text{otherwise}. \end{cases} \]

\[ h_0^{(e)} = \tanh(W_{\overrightarrow{f}} \overrightarrow{h}_{|F|} + W_{\overleftarrow{f}} \overleftarrow{h}_1 + b_e) \]

Motivation:
- Help bootstrap learning
- By shortening length of dependencies

Motivation:
- Take 2 hidden vectors from source encoder
- Combine them into a vector of size required by decoder
Introduction to Neural Machine Translation

• Neural language models review

• Sequence to sequence models for MT
  • Encoder-Decoder
  • Sampling and search (greedy vs beam search)
  • Practical tricks

• Sequence to sequence models for other NLP tasks

• Attention mechanism