TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2019

Language Modeling

Machine Translation

Attention

Beam Search

Error-Based Training

RNN for a generic CELL Procedure

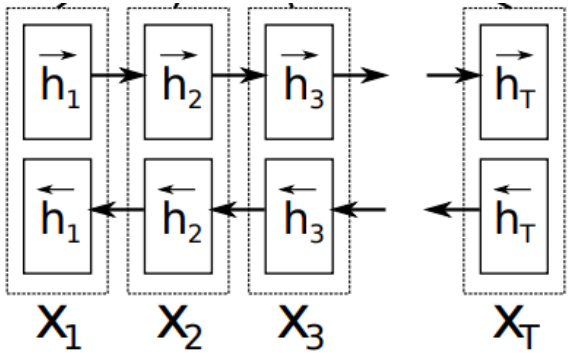
As usual, we use capital letter indeces to denote whole tensors or slices and lower case letters to denote particular index values.

Procedure $\mathrm{RNN}_{\Phi}(x(T, I))$

$$
\begin{aligned}
h[0, J] &= \mathrm{CELL}_{\Phi.\mathrm{cell}}(\Phi.\mathrm{init}[J],\ x[0, I]) \\
\text{for } t > 0 \quad h[t, J] &= \mathrm{CELL}_{\Phi.\mathrm{cell}}(h[t-1, J],\ x[t, I])
\end{aligned}
$$

Return $h[T, J]$

bi-directional RNNS

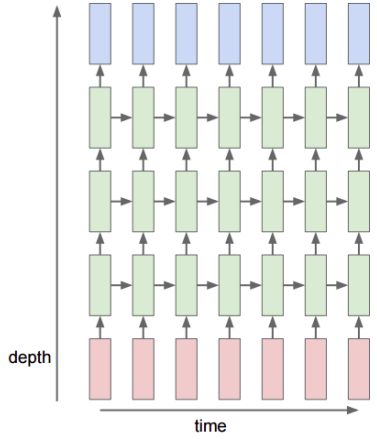


$$\vec{h}[T, J] = \vec{\mathrm{RNN}}_{\Phi.LR}(x[T, I])$$

$$\overleftarrow{h}[T, J] = \overleftarrow{\mathrm{RNN}}_{\Phi.RL}(x[T, I])$$

for t $h[t, 2J] = \vec{h}[t, J]; \overleftarrow{h}[t, J]$ where $x; y$ is vector concatenation

Multi-Layer RNNs



[Figure by Leonardo Araujo dos Santos]

$$h[0, T, J] = \mathrm{RNN}_{\Phi[0]}(x[T, I])$$

$$\text{for } \ell > 0 \quad h[\ell, T, J] = \mathrm{RNN}_{\Phi[\ell]}(h[\ell - 1, T, J])$$

Each layer can be bidirectional.

Residual Multi-Layer RNNs

$$h[0, T, J] = \mathrm{RNN}_{\Phi[0]}(x[T, I])$$

$$\text{for } \ell > 0 \quad h[\ell, T, J] = h[\ell - 1, T, J] + \mathrm{RNN}_{\Phi[\ell]}(h[\ell - 1, T, J])$$

This is used in Google translation.

Language Modeling

Let W be some finite vocabulary of tokens (words).

Let Pop be a population distribution over W^* (sentences).

We will write a sequence $w[1], \ldots, w[t]$ as $w[T]$.

We want to train a model $P_\Phi(w[T])$.

$$\Phi^* = \operatorname*{argmin}_\Phi \; E_{\text{Pop}} \; - \ln P_\Phi(w[T])$$

The End of Sentence Token

We assume a special toeken `<EOS>` called the end of sentence token.

Let t_{final} be the last time index allowed for T.

We requite $w[t_{\text{final}}] =$ `<EOS>` and $w[t] \neq$ `<EOS>` for $t < t_{\text{final}}$.

This gives:

$$P(w[T]) = \prod_t P(w[t] \mid w[1], \ldots, w[t-1])$$

Autoregressive Models are Friendly

I will call a model $P_\Phi(y)$ "friendly" if we can efficiently sample from P_Φ and we can also efficiently compute $P_\Phi(y)$ for any given y.

Multiclass classification models for small K are friendly.

Gaussian distributions are friendly.

Autoregressive language models are friendly — they are a special case of friendly graphical models.

General graphical models are often unfriendly.

Word Embeddings

We will use a word embedding tensor $e[W, I]$ which should be interpreted as assigning a vector $e[w, I]$ to each word w.

The word embedding tensor $e[W, I]$ is a parameter of language models (and many other kinds of NLP models).

Autoregressive Language Modeling

Procedure $P_\Phi(w[T])$;; $w[T]$ given

$$x[t, I] = (\Phi.\text{embed})[w[t], I]$$

$$h[T, J] = \text{RNN}_{\Phi.\text{RNN}}(x[T, J]) \quad J \geq I$$

$$s[0, w] = (\Phi.\text{embed}[w, I]) \cdot (\Phi.\text{init}[I])$$

$$\text{for } t > 0 \quad s[t, w] = (\Phi.\text{embed}[w, I]) \cdot h[t-1, I] \quad h \text{ truncated to } I$$

$$p[t, w] = \underset{w}{\text{softmax}} \; s[t, w]$$

Return $\prod_t p[t, w[t]]$

Language Model Loss Decomposition

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} \; E_{\mathrm{Pop}} \; -\ln \; P_\Phi(w[T])$$

$$= \underset{\Phi}{\operatorname{argmin}} \; E_{\mathrm{Pop}} \sum_t \; -\ln \; p[t, w[t]]$$

Standard Measures of Performance

Bits per Character: For character language models performance is measured in bits per character. Typical numbers are slightly over one bit per character.

Perplexity: It would be natural to measure word language models in bits per word. However, it is traditional to measure them in perplexity which is defined to be 2^b where b is bits per word. Perplexities of about 60 are typical.

According to Quora there are 4.79 letters per word. 1 bit per character (including space characters) gives a perplexity of $2^{5.79}$ or 55.3.

Sampling From an Autoregressive Model

To sample a sentence

$$w_1, \ldots, w_T, \texttt{<eos>}$$

we sample w_t from

$$P_\Phi(w_t | w_1, \ldots, w_{t-1})$$

until we get <eos>.

Machine Translation

$$w_1^{\text{in}}, \ldots, w_{t_{\text{in}}}^{\text{in}} \Rightarrow w_1^{\text{out}}, \ldots, w_{t_{\text{out}}}^{\text{out}}$$

$$w_{\text{in}}[T_{\text{in}}] \Rightarrow w_{\text{out}}[T_{\text{out}}]$$

Translation is a **sequence to sequence** (seq2seq) task.

Sequence to Sequence Learning with Neural Networks, Sutskever, Vinyals and Le, NIPS 2014, arXiv Sept 10, 2014.

Machine Translation

$$w_{\text{in}}[T_{\text{in}}] \Rightarrow w_{\text{out}}[T_{\text{out}}]$$

We define a model

$$P_\Phi \left(w_{\text{out}}[T_{\text{out}}] \mid w_{\text{in}}[T_{\text{in}}] \right)$$

$$\begin{aligned} \Phi^* &= \underset{\Phi}{\text{argmin}} \; E_{\text{Pop}} \; - \ln \; P_\Phi(w_{\text{out}}[T_{\text{out}}] \mid w_{\text{in}}[T_{\text{in}}]) \\ &= \underset{\Phi}{\text{argmin}} \; E_{\text{Pop}} \; - \ln P_\Phi(y|x) \end{aligned}$$

A Simple RNN Translation Model

We construct a conditional language model

$$\text{Init}[J] = \overleftarrow{\text{RNN}}_{\Phi.\text{in}}(w_{\text{in}}[T_{\text{in}}])\ [0, J]$$

$$P_\Phi(w_{\text{out}}[T_{\text{out}}] \mid w_{\text{in}}[T_{\text{in}}]) = P_{\Phi.\text{out}}(w_{\text{out}}[T_{\text{out}}] \mid \text{Init}[J])$$

Here $\overleftarrow{\text{RNN}}_{\Phi.\text{in}}(w_{\text{in}}[T_{\text{in}}])\ [0, J]$ is used as the initial hidden state of an RNN language model for the output.

$\overleftarrow{\text{RNN}}_{\Phi.\text{in}}(w_{\text{in}}[T_{\text{in}}])\ [0, J]$ is a "thought vector" representation of the input sentence.

Reverse Order Encoding

Having the encoding run in the reverse order of the decoding helps in training by making it easier to start the translation correctly (assuming that the languages have similar word orders).

In the original paper the encoding was done left to right and the decoding was done right to left.

Machine Translation Decoding

We can sample from $P_{\Phi.\text{out}}(w[T] \mid H[J])$.

But we might prefer

$$w_{\text{out}}[T_{\text{out}}] = \underset{w_{\text{out}}[T_{\text{out}}]}{\text{argmax}} \; P_{\Phi}\left(w_{\text{out}}[T_{\text{out}}] \mid w_{\text{in}}[T_{\text{in}}]\right)$$

This is typically approximated with greedy decoding:

$$w_{\text{out}}[t+1] = \underset{w}{\text{argmax}} \; p_{\text{out}}[t+1, w]$$

These are not the same.

Attention-Based Translation

Neural Machine Translation by Jointly Learning to Align and Translate Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio, ICLR 2015 (arXiv Sept. 1, 2014)

There are many ways to construct attention-based translation models.

Details are often unimportant.

The model presented in these slides has been optimized for simplicity.

Representing Sentences by Vector Sequences

The input sentence is now represented by a sequence of vectors.

$$P_{\Phi}(w_{\text{out}}[T_{\text{out}}] \mid w_{\text{in}}[T_{\text{in}}])$$

$$= P_{\Phi.\text{out}}(w_{\text{out}}[T_{\text{out}}] \mid \overleftarrow{\text{RNN}}_{\Phi.\text{RNN}}(w_{\text{in}}[T_{\text{in}}]))$$

We still use $\overleftarrow{\text{RNN}}_{\Phi.\text{RNN}}(w_{\text{in}}[T_{\text{in}}])\ [0, J]$ as the initial state of a decoding RNN.

But the decoding RNN now has access to the entire sequence of hidden states for the input.

Attention-Based Translation

Memory-Based Language Modeling

Procedure $P_\Phi(w[T] \mid M[T_M, J])$;;$w[T]$ given

$$
\begin{aligned}
x[t, I] &= (\Phi.\text{embed})[w[t], I] \\
h[T, J] &= \text{RNN}_{\Phi.\text{RNN}}(x[T, I] \mid M[T_M, J]) \quad J \geq I \\
s[0, w] &= (\Phi.\text{embed}[w, I]) \cdot (\Phi.\text{init}[I]) \\
\text{for } t > 0 \quad s[t, w] &= (\Phi.\text{embed}[w, I]) \cdot h[t-1, I] \quad h \text{ truncated to } I \\
p[t, w] &= \underset{w}{\text{softmax}} \; s[t, w]
\end{aligned}
$$

Return $\prod_t p[t, w[t]]$

Attention-Based (Memory-Based) Conditional RNN

Procedure $\text{RNN}_\Phi(x[T_x, I] \mid M[T_M, J])$

$$h[0, J] = \text{CELL}_\Phi(M[0, J]; x[t, I]; 0[J])$$

for $t > 0$

$$h[t, J] = \text{CELL}_\Phi(\ h[t-1, J]; x[t, I]; \text{Lookup}(h[t-1, J], M[T, J])\ )$$

Return $h[T, I]$

Attention as a Key-Value Memory Mechanism

Procedure Lookup($\text{key}[J],\ M[T, J]$)

$$s[t] = \text{key}[J]^\top M[t, J]$$

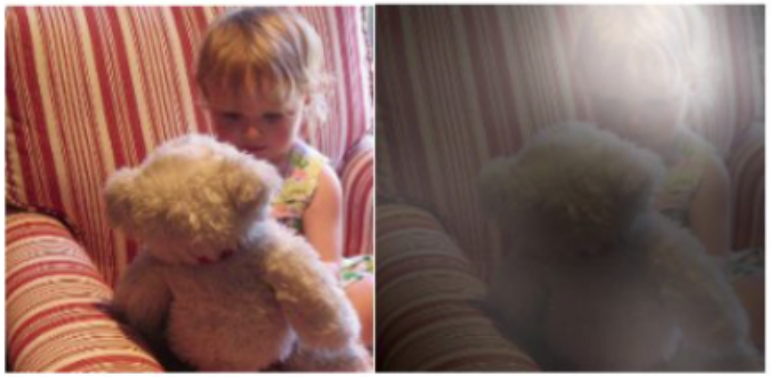
$$\alpha[t] = \underset{t}{\text{softmax}}\ s[t]\ \text{; the attention}$$

$$V[J] = \sum_t \alpha[t] M[t, J]$$

Return $V[J]$

Attention in Image Captioning

A woman is throwing a frisbee in a park.



A little girl sitting on a bed with a teddy bear.

Xu et al. ICML 2015

Greedy Decoding vs. Beam Search

We would like

$$W_{\text{out}}[T_{\text{out}}]^* = \underset{W_{\text{out}}[T_{\text{out}}]}{\text{argmax}}\ P_\Phi(W_{\text{out}}[T_{\text{out}}] \mid W_{\text{in}}[T_{\text{in}}])$$

But a greedy algorithm may do well

$$w_t = \underset{w}{\text{argmax}}\ P_\Phi(w \mid W_{\text{in}}[T_{\text{in}}],\ w_1, \ldots, w_{t-1})$$

But these are not the same.

Example

"Those apples are good" vs. "Apples are good"

$$P_\Phi(\text{Apples are Good } \texttt{<eos>}) > P_\Phi(\text{Those apples are good } \texttt{<eos>})$$

$$P_\Phi(\text{Those}|\varepsilon) > P_\Phi(\text{Apples}|\varepsilon)$$

Beam Search

At each time step we maintain a list the K best words and their associated hidden vectors.

This can be used to produce a list of k "best" decodings which can then be compared to select the most likely one.

Phrase Based Statistical Machine Translation (SMT)

Step I: Learn a phrase table — a set of triples (p, q, s) where

- p is a (short) sequence of source words.
- q is a (short) sequence of target words.
- s is a score.

("au", "to the", .5) ("au banque", "for the bank", .01)

For a phrase triple P we will write P.source for the source phrase, P.target for the target phrase, and P.score for the score.

Derivations

Consider an input sentence x of length T.

We will write $x[s : t]$ for the substring $x[s]$, ..., $x[t-1]$.

A derivation d from x is a sequence $(P_1, s_1, t_1,), \ldots, (P_K, s_K, t_K)$ where P_k.source $= x[s_k : t_k]$.

The substrings $x[s_k : t_k]$ should be disjoint and "cover" x.

For $d = [(P_1, s_1, t_1,), \ldots, (P_L, s_K, t_K)]$ we define

$$y(d) \equiv P_1.\text{target } \cdots P_K.\text{target}$$

We let $D(x)$ be the set of derivations from x.

Scoring

For $d \in D(x)$ we define a score $s(d)$

$$s(d) = \alpha \ln P_{\mathrm{LM}}(y(d)) + \beta \sum_k P_k.\mathrm{score} + \gamma\ \mathrm{distortion}(d)$$

where $P_{\mathrm{LM}}(y)$ is the probability assigned to string y under a language model for the target language

and distortion(d) is a measure of consistency of word ordering between source and target strings as defined by the indeces $(s_1, t_1), \ldots, (s_K, t_K)$.

Translation

$$y(x) = y(d^*(x))$$

$$d^*(x) = \underset{d \in D(x)}{\text{argmax}} \; s(d)$$

END