TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2018

Language Modeling

Machine Translation

Attention

Beam Search

Error-Based Training

Language Modeling

We are given a sequence $x^1, \ldots, x^T$ and we want to compute a model probability.

$$Q_\Phi(x_1, \ldots, x^T)$$

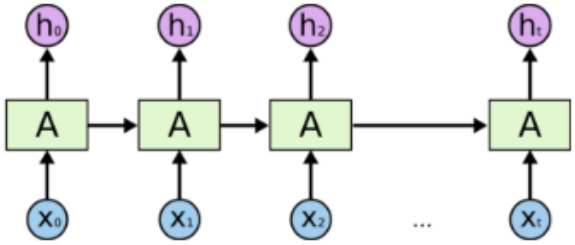
Here x^t can be either characters or words.

We represent the probability as a product of conditionals.

$$Q_\Phi(x_1, \ldots, x^T) = \prod_t Q_\Phi(x^t \mid x^1, \ldots, x^{t-1})$$

A model of this form is called **Autoregressive** — a model (regression) is used to predict the next element from the previous elements.

RNN Language Modeling



[Christopher Olah]

$$Q_\Phi(x^{t+1} \mid x^1, \ldots, x^t) = \text{Softmax}(W^o\, h^t)$$

For word language models this softmax is the computation bottleneck in training.

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 then in perplexity which is defined to be 2^b where b is bits per world. 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

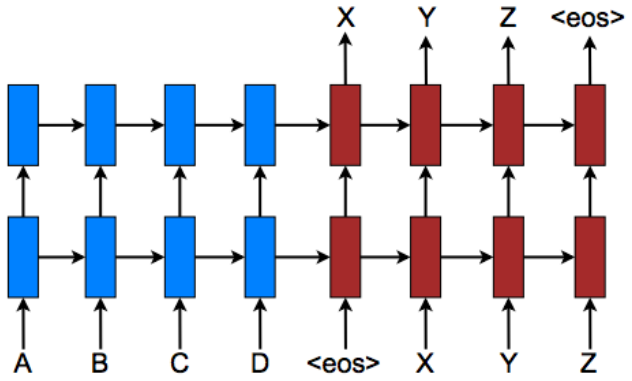
$$x^1, \ldots, x^T, \texttt{<eos>}$$

we sample x^t from

$$Q_\Phi(x^t | x^1, \ldots, x^{t-1})$$

until we get <eos>.

Machine Translation



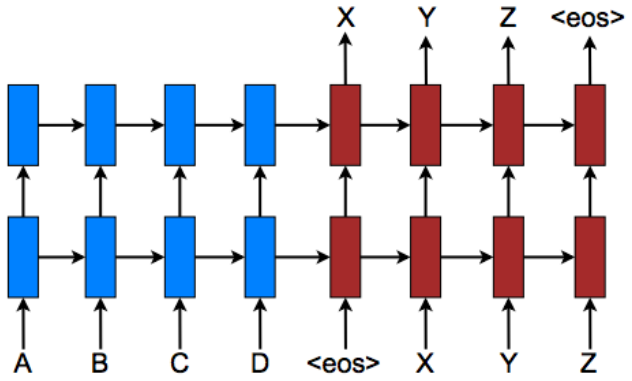
[Figure from Luong et al.]

$$DCBA \Rightarrow XYX$$

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.

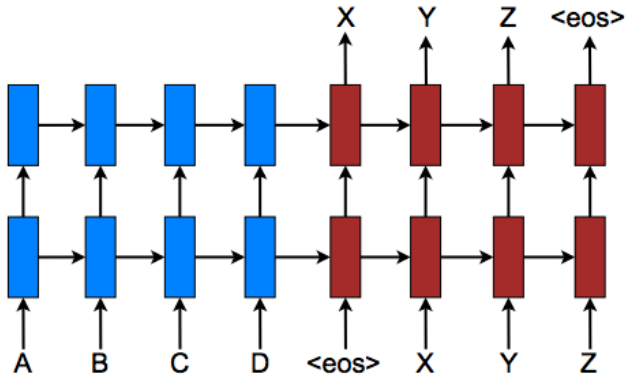
Machine Translation



[Figure from Luong et al.]

The input sentence is represented by a "thought vector" produced by an RNN.

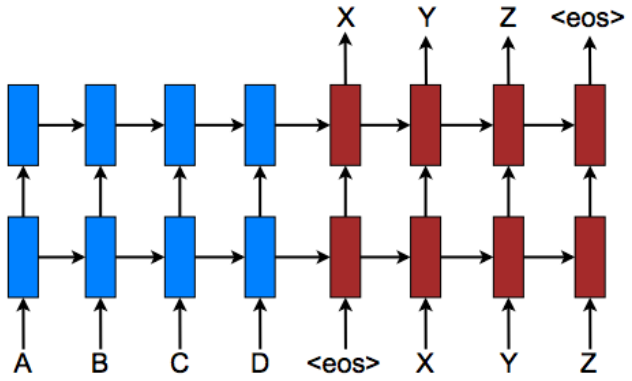
Machine Translation



[Figure from Luong et al.]

The thought vector is the initial state vector used in "sampling" a translation.

Machine Translation

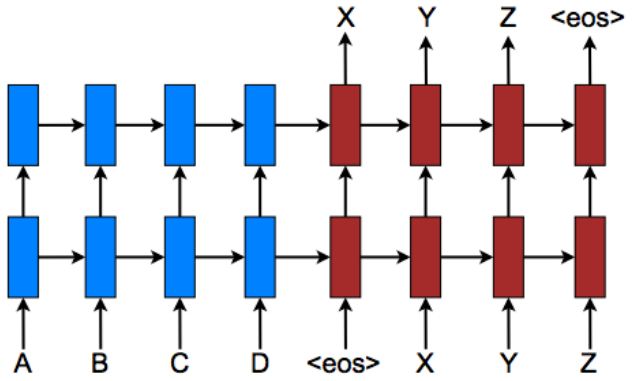


[Figure from Luong et al.]

For the training pair $DCBA \Rightarrow XYX$ the training loss is

$$-\log\ Q_\Phi(XYZ \mid DCBA)$$

Machine Translation



[Figure from Luong et al.]

In Sutskever et al. (2014) the LSTMs are layered 4 deep.

Attention-Based Translation

Translation is improved by aligning input words with output words.

This is done with "Attention".

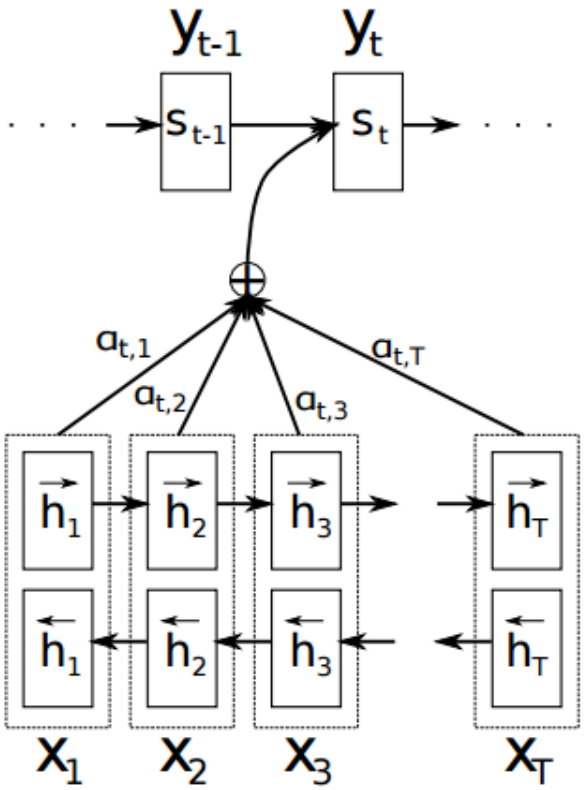
Neural Machine Translation by Jointly Learning to Align and Translate Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio, ICLR 2015 (arXiv Sept. 1, 2014)

Attention

The input sentence is no longer represented by just one thought vector.

Instead the entire sequence of hidden vectors produced by the RNN is used during generation of the translation.

Attention



[Bahdanau, Cho, Bengio (2014)]

A first step: BiRNNs

$$\vec{h}^{t+1} = \text{RNNCELL}_{\Phi}(\vec{h}^{t}, x^{t})$$

$$\overleftarrow{h}^{t-1} = \text{RNNCELL}_{\Psi}(\overleftarrow{h}^{t}, x^{t})$$

$$\overleftrightarrow{h}^{t} = \left[\vec{h}^{t}, \overleftarrow{h}^{t}\right] \qquad ([x, y] \text{ denotes vector concatenation})$$

Basic Sequence to Sequence Model

$$
\begin{aligned}
s^0 &= \vec{h}^T \\
y^0 &= \text{<eos>}
\end{aligned}
$$

$$
\begin{aligned}
Q^{\Phi}(:\ |\mathbf{x},\ y^1, \ldots, y^i) &= \underset{\hat{y}}{\text{softmax}}\ W^y\ s^{i+1} \\
s^{i+1} &= \text{RNNCELL}_{\Phi}(s^i,\ e(y^i)))
\end{aligned}
$$

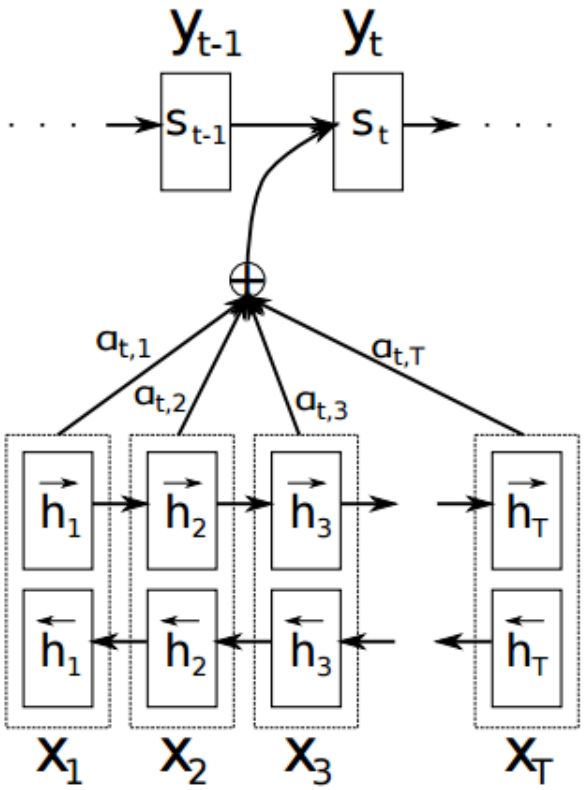
Adding Attention

$$
\begin{aligned}
c^0 &= \overleftrightarrow{h}^T \\
s^0 &= \vec{h}^T \\
y^0 &= \text{<eos>}
\end{aligned}
$$

$$
\begin{aligned}
Q_\Phi(: \ |\mathbf{x}, \ y^1, \ldots, y^i) &= \underset{\hat{y}}{\text{softmax}} \ W^y \ s^{i+1} \\
s^{i+1} &= \text{RNNCELL}_\Phi(s^i, \ [e(y^i), \ c^i])
\end{aligned}
$$

$$
\begin{aligned}
c^i &= \sum_t \alpha^{i,t} \ \overleftrightarrow{h}^t \\
\alpha^{i,t} &= \underset{t}{\text{softmax}} \ \tanh(W^a \ [s^i, \ \overleftrightarrow{h}^t])
\end{aligned}
$$

Attention



[Bahdanau, Cho, Bengio (2014)]

Bilinear Attention is More Popular Today

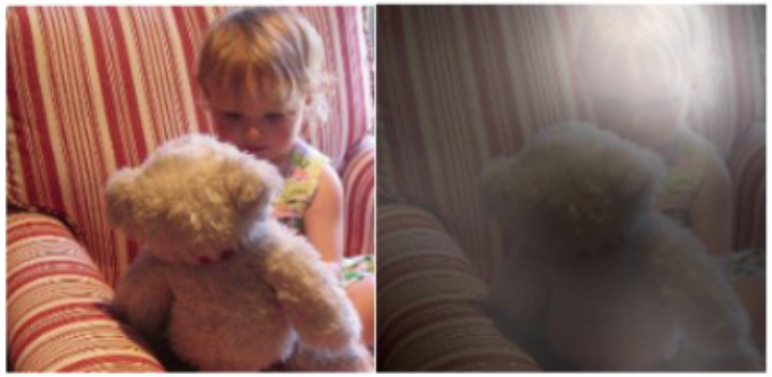
$$
\begin{aligned}
c^0 &= \overleftrightarrow{h}^T \\
s^0 &= \vec{h}^T \\
y^0 &= \text{<eos>}
\end{aligned}
$$

$$
\begin{aligned}
Q_\Phi(: \ |\mathbf{x},\ y^1, \ldots, y^i) &= \underset{\hat{y}}{\text{softmax}}\ W^y\ s^{i+1} \\
s^{i+1} &= \text{RNNCELL}_\Phi(s^i,\ [e(y^i),\ c^i])
\end{aligned}
$$

$$
\begin{aligned}
c^i &= \sum^t \alpha^{i,t}\ \overleftrightarrow{h}^t \\
\alpha^{i,t} &= \underset{t}{\text{softmax}}\ (s^i)^\top W^a\ \overleftrightarrow{h}^t
\end{aligned}
$$

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

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmax}}\, Q_\Phi(\mathbf{y}|\mathbf{x})$$

A **greedy algorithm** may do well

$$y^{t+1} = \underset{\hat{y}}{\operatorname{argmax}}\ Q_\Phi(\hat{y} \mid \mathbf{x},\ y^1, \ldots, y^t)$$

But these are not the same.

Example

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmax}}\, Q_\Phi(\mathbf{y}|\mathbf{x})$$

$$y^{t+1} = \underset{\hat{y}}{\operatorname{argmax}}\; Q_\Phi(\hat{y} \mid \mathbf{x},\; y^1, \ldots, y^t)$$

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

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

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

Beam Search

At each time step we maintain a list of k word-vector pairs:

$$Y^t = ((y^{t,1}, h^{t,1}), \ldots, (y^{t,k}, h^{t,k})$$

$$Y^{t+1} = \underset{(\hat{y},\hat{h})\in C^t}{\text{kbest}} \; Q_\Phi(y \mid \mathbf{x},\, Y^1, \ldots, Y^t)$$

$$C^t = \left\{ (y^{t+1,i,j}, h^{t+1,i}) : \begin{array}{l} y^{t+1,i,j} \in \text{kbest}_{\hat{y}}\; Q_\Phi(\hat{y}|(y^{t,i}, h^{t,i})) \\ \\ h^{t+1,i,j} = \text{RNNCELL}(h^{t,i}, e(y^{t,i,j})) \end{array} \right\}$$

Error-Based Training

Systems are often evaluated by error rather than loss (log loss).

Should we train directly to minimize error?

Should translation be trained directly on BLEU score?

Should segmentation be trained on intersection over union?

Should cancer screening be trained on recall?

If the model provides probabilities we can do Bayesian inference. But the model might not be sufficiently expressive.

Label Adjustment

We can consider an arbitrary error function $\mathrm{Err}(y, \hat{y})$ assigning an error value the true label is y and the system guesses $\hat{y}$.

$$
\begin{aligned}
f(\hat{y}) &= \sum_{\alpha} f[\alpha, \hat{y}[\alpha]] \\
\hat{y}^*(f) &= \operatorname*{argmax}_{\hat{y}} \; f(\hat{y}) \\
\breve{y}^*(f) &= \operatorname*{argmax}_{\breve{y}} \; f(\breve{y}) - \epsilon \mathrm{Err}(y, \breve{y}) \quad \textbf{(adjusted label)} \\
\mathrm{error}(y, f) &= \mathrm{Err}(y, \hat{y}^*(f))
\end{aligned}
$$

$$
f_\Phi(x).\mathrm{grad}[\alpha, \tilde{y}] \texttt{ -= } \eta \left(\mathbb{1}[\hat{y}^*_{f_\Phi(x)}[\alpha] = \tilde{y}] - \mathbb{1}[\breve{y}^*_{f_\Phi(x)}[\alpha] = \tilde{y}] \right)
$$

Label Adjustment Theorem

$$f_\Phi(x).\text{grad}[\alpha, \tilde{y}] \texttt{ -= } \eta \left(\mathbb{1}[\hat{y}^*_{f_\Phi(x)}[\alpha] = \tilde{y}] - \mathbb{1}[\breve{y}^*_{f_\Phi(x)}[\alpha] = \tilde{y}] \right)$$

Theorem: For a continuous and smooth population distribution

$$\nabla_\Phi \; E_{(x,y)\sim\text{Pop}} \; \text{Err}(y, \hat{y}^*(f_\Phi(x)))$$

$$= \lim_{\epsilon \to 0} \; \frac{1}{\epsilon} \; E_{(x,y)\sim\text{Pop}}$$

$$\sum_{\alpha,\tilde{y}} \left(\; \mathbb{1}[\hat{y}^*_{f_\Phi(x)}[\alpha] = \tilde{y}] - \mathbb{1}[\breve{y}^*_{f_\Phi(x)}[\alpha] = \tilde{y}] \right) \nabla_\Phi f_\Phi(x)[\alpha, \tilde{y}]$$

Intersection over Union

In visual detection problems one is typically evaluated by **intersection over union**.

$$\textbf{IOU} = \frac{|\text{true positives} \cap \text{false positives}|}{|\text{true positives} \cup \text{false positives}|} = \frac{P - FN}{P + FP}$$

$$\frac{\partial IOU}{\partial \text{FP}} = \frac{-(P - FN)}{(P + FP)^2} = \frac{-\text{IOU}}{P + FP}$$

$$\frac{\partial IOU}{\partial \text{FN}} = \frac{-1}{P + FP}$$

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", "the the bank", .01)

For a phrase 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