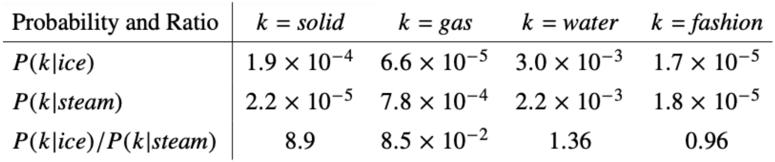
Advanced Machine Learning, Lecture 9

András Kornai

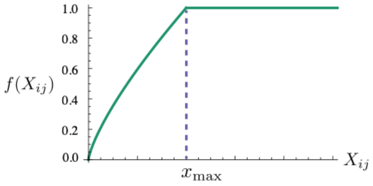
BME 2020 Nov 5

Back to word vectors

- Refresher from Lecture 5: static embeddings are obtained by low-rank approximation of positive pointwise mutual information (PPMI) matrix
- But this was understood only after several years (by Levy and Goldberg 2014a)
- A good explanation of the philosophy behind the earlier optimization is Pennington et al 2014
- Let X_{ij} count the number of times j occurs in the context of i
- $\frac{X_{ij}}{\sum_k X_{ik}} = P(j|i)$. We really care about the ratios:
- 

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

GloVe

- So we want some model function $F(w_i, w_j, \tilde{w}_k) = P_{ik}/P_{jk}$
- Inside F, we may as well go linear: $F((w_i - w_j)^T \tilde{w}_k) = P_{ik}/P_{jk}$
- Great trick: make sure $F(w_i, w_j, \tilde{w}_k) = F(w_i^T \tilde{w}_k)/F(w_j^T \tilde{w}_k)$
- This forces F=exp. Would yield $w_i^T \tilde{w}_k = \log P_{ik} = \log X_{ik} - \log \sum_k X_{ik}$ but we absorb the marginal in the bias, to obtain
- $w_i^T \tilde{w}_k + b_i - \tilde{b}_k = \log X_{ik}$
- To deal with issues of too extreme values, a dampening function $(x/x_m)^{3/4}$ for $x \leq x_m$, $f = 1$ for $x \geq x_m$ is used to minimize the weighted quadratic sum $\sum_{i,j}^V f(X_{ij})(w_i^T \tilde{w}_k + b_i - \tilde{b}_k - \log X_{ik})^2$
- 

GloVe and skip-gram

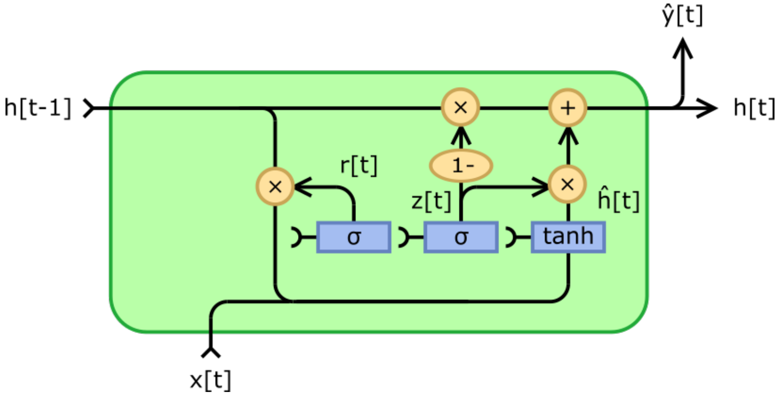
- Let Q_{ij} be the softmax estimate of $P(i|j)$ given by $\exp w_i^T \tilde{w}_j / \sum_{k=1}^V \exp w_i^T \tilde{w}_k$
- The objective is $J = \sum_{i=1}^V X_i H(P_i, Q_i)$ where $H(P_i, Q_i)$ is the cross-entropy of the distributions P_i and Q_i
- We may as well look at a least squares model minimizing $\hat{J} = \sum_{i,j} X_i (X_{ij} - \exp w_i^T \tilde{w}_j)$
- Key takeway from `Pennington et al 2014`: "In fact, it is possible to optimize (for J) directly as opposed to the on-line training methods used in the (neural) models"
- Lesson: neural nets (which perform incremental optimization) are good heuristics, but not as good models as the globally optimized ones
- This story may repeat for dynamic embeddings
- These handle time in various ways. Early variants (Jordan 1986) don't scale well

Recursive Neural Networks

- The classic RNN definition (Elman 1990) has a time-dependent state vector h_t, and upon receiving input x_t moves to h_{t+1} given by an affine transform and a tanh nonlinearity: $h_{t+1} = \tanh Wx_t + Uh_t + b$ where the matrices W, U and the bias vector b are the trainable parameters of the model, (Jordan 1986 used the output from previous stage in update)
- Classic result (Siegelmann and Sontag 1992): RNNs can simulate any TM in real time using only rational weights, we can even build a universal TM from ~ 1000 very simple processors
- But this requires arbitrary precision arithmetic, and problem instance is read into machine ahead of time (and may require superexponential number of processing steps afterwards). Practical interest is with finite precision and real-time operation
- Variant of Elman model uses ReLU: max(0,x) instead of tanh – computationally more powerful, but hard to train 'exploding gradients'

GRU (gated recurrent unit)

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{1}$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \tag{2}$$
$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{3}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \tag{4}$$

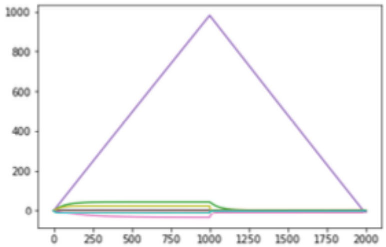


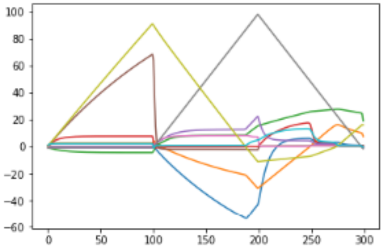
LSTM (long short-term memory)

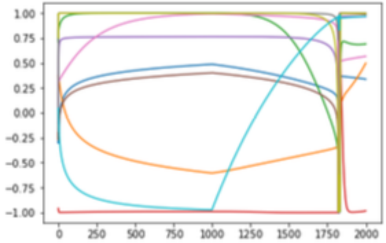
$$\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) && (5)\\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) && (6)\\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) && (7)\\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) && (8)\\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t && (9)\\
h_t &= o_t \circ \sigma_h(c_t) && (10)
\end{aligned}$$

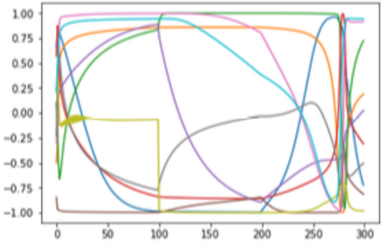
- Historically LSTM (Hochreiter and Schmidhuber 1996) preceded GRU (Cho et al 2014) who wanted to simplify LSTMs
- LSTMs have more power (do better on long dependencies)
- Computational power of architectures investigated by Weiss et al (2018)
- Key idea: keep a very contentful state vector
- Possible line of attack: information bottleneck method (Tishby et al 1999)

LSTM v GRU

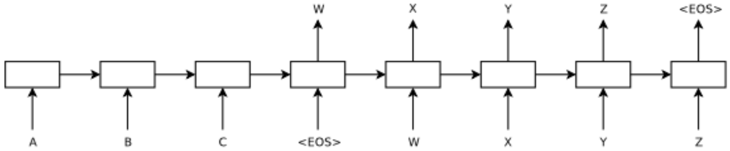

(a) $a^n b^n$-LSTM on $a^{1000} b^{1000}$


(b) $a^n b^n c^n$-LSTM on $a^{100} b^{100} c^{100}$


(c) $a^n b^n$-GRU on $a^{1000} b^{1000}$


(d) $a^n b^n c^n$-GRU on $a^{100} b^{100} c^{100}$

Seq2seq

- Using LSTMs as elementary building blocks (Sutskever et al 2014)
- 

- Stacked 5 deep, state vectors 8000 dim
- Does MT (English-French) quite well
- Relies on reversing input
- Encoder-decoder architecture (can be retrojected on LSTM)

Attention

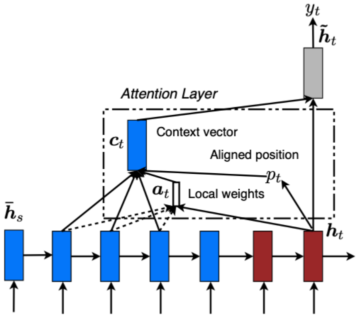
- Bahdanau et al 2015, Luong at al 2015
- 

  Figure 3: **Local attention model** – the model first predicts a single aligned position p_t for the current target word. A window centered around the source position p_t is then used to compute a context vector c_t, a weighted average of the source hidden states in the window. The weights a_t are inferred from the current target state h_t and those source states $\bar{h}_s$ in the window.
- Self-attention Lin 2017 Vaswani et al 2017
- Also developed for MT (which remains the canonical case)
- Introduces 'multi-head' model: several attention layers running in parallel
- Positional encoding: mixing sinusoids of different frequencies with the input

Seq2seq + Attention = Transformers

- The first dynamic word vector system was CoVe (McCann 2017)
- This was an encoder-decoder model trained on various MT datasets (but no effort to mix them in a single model)
- Trained on MT data (7m sentence pairs)
- Encoder output concatenated to a static (GloVe) embedding
- CoVe had sophisticated bidirectional attention, but not as good as Transformers

ELMO

- Encoder-decoder trained on LM task (monolingual – much more data)
- Multi-head (transformer-style) attention
- Concatenates all (not just the top) LSTM states
- For specific tasks, it may make sense to re-train the LM itself
- ELMO training used 1G words of English text, GPT-2 on about 8G words, GPT-3 on over 100G words (45 TB compressed from CommonCrawl, plus curated datasets)
- GPT-3 175G parameters trained in $3.14 \cdot 10^{23}$ flops (a third yottaflop)
- Energy usage alone 500MWh

BERT

- Introduced in Devlin et al (2019)
- Similar to ELMO, but trained on much less data than GPT: 800m words from the Google Books Corpus and 2.5G words from WP
- Fully bidirectional, with 15% of tokens masked out
- m-BERT (multilingual, 104 languages), RoBertA, etc etc
- National BERT's: CememBERT (Martin et al 2019), HuBERT (Nemeskey 2020), ...
- Generalizations, BERTology