# Recurrent Networks, and LSTMs, for NLP

Michael Collins, Columbia University

# Representing Sequences

- Often we want to map some sequence $x_{[1:n]} = x_1 \ldots x_n$ to a label $y$ or a distribution $p(y|x_{[1:n]})$
- Examples:
  - Language modeling: $x_{[1:n]}$ is first $n$ words in a document, $y$ is the $(n+1)$'th word
  - Sentiment analysis: $x_{[1:n]}$ is a sentence (or document), $y$ is label indicating whether the sentence is positive/neutral/negative about a particular topic (e.g., a particular restaurant)
  - Machine translation: $x_{[1:n]}$ is a source-language sentence, $y$ is a target language sentence (or the first word in the target language sentence)

# Representing Sequences (continued)

- Slightly more generally: map a sequence $x_{[1:n]}$ **and a position** $i \in \{1 \ldots n\}$ to a label $y$ or a distribution $p(y|x_{[1:n]}, i)$
- Examples:
  - Tagging: $x_{[1:n]}$ is a sentence, $i$ is a position in the sentence, $y$ is the tag for position $i$
  - Dependency parsing: $x_{[1:n]}$ is a sentence, $i$ is a position in the sentence, $y \in \{1 \ldots n\}, y \neq i$ is the head for word $x_i$ in the dependency parse

# A Simple Recurrent Network

**Inputs:** A sequence $x_1 \dots x_n$ where each $x_j \in \mathbb{R}^d$. A label $y \in \{1 \dots K\}$. An integer $m$ defining size of hidden dimension. Parameters $W^{hh} \in \mathbb{R}^{m \times m}$, $W^{hx} \in \mathbb{R}^{m \times d}$, $b^h \in \mathbb{R}^m$, $h^0 \in \mathbb{R}^m$, $V \in \mathbb{R}^{K \times m}$, $\gamma \in \mathbb{R}^K$. Transfer function $g : \mathbb{R}^m \rightarrow \mathbb{R}^m$.
**Definitions:**

$$\begin{aligned} \theta &= \{W^{hh}, W^{hx}, b^h, h^0\} \\ R(x^{(t)}, h^{(t-1)}; \theta) &= g(W^{hx}x^{(t)} + W^{hh}h^{(t-1)} + b^h) \end{aligned}$$

**Computational Graph:**

- For $t = 1 \dots n$
  - $h^{(t)} = R(x^{(t)}, h^{(t-1)}; \theta)$
- $l = Vh^{(n)} + \gamma$, $q = \text{LS}(l)$, $o = -q_y$

# The Computational Graph

# A Problem in Training: Exploding and Vanishing Gradients

- Calculation of gradients involves multiplication of long chains of Jacobians
- This leads to exploding and vanishing gradients

# LSTMs (Long Short-Term Memory units)

- **Old definitions of the recurrent update:**

$$\begin{aligned}\theta &= \{W^{hh}, W^{hx}, b^h, h^0\} \\ R(x^{(t)}, h^{(t-1)}; \theta) &= g(W^{hx}x^{(t)} + W^{hh}h^{(t-1)} + b^h)\end{aligned}$$

- LSTMs give an alternative definition of $R(x^{(t)}, h^{(t-1)}; \theta)$.

# Definition of Sigmoid Function, Element-Wise Product

- Given any integer $d \geq 1$, $\sigma^d : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is the function that maps a vector $v$ to a vector $\sigma^d(v)$ such that for $i = 1 \dots d$,

$$\sigma_i^d(v) = \frac{e^{v_i}}{1 + e^{v_i}}$$

- Given vectors $a \in \mathbb{R}^d$ and $b \in \mathbb{R}^d$, $c = a \odot b$ has components

$$c_i = a_i \times b_i$$

  for $i = 1 \dots d$

# LSTM Equations (from Ilya Sutskever, PhD thesis)

Maintain $s^t, \tilde{s}^t, h^t$ as hidden state at position $t$. $s^t$ is *memory*, intuitively allows long-term memory. The function $s^t, \tilde{s}^t, h^t = \text{LSTM}(x^t, s^{t-1}, \tilde{s}^{t-1}, h^{t-1}; \theta)$ is defined as:

$$
\begin{aligned}
u^t &= \text{CONCAT}(h^{t-1}, x^t, \tilde{s}^{t-1}) \\
h^t &= g(W^h u^t + b^h) \quad \text{(hidden state)} \\
i^t &= g(W^i u^t + b^i) \quad \text{(“input”)}
\end{aligned}
$$

$$
\begin{aligned}
\iota^t &= \sigma(W^\iota u^t + b^\iota) \quad \text{(“input gate”)} \\
o^t &= \sigma(W^o u^t + b^o) \quad \text{(“output gate”)} \\
f^t &= \sigma(W^f u^t + b^f) \quad \text{(“forget gate”)}
\end{aligned}
$$

$$
\begin{aligned}
s^t &= s^{t-1} \odot f^t + i^t \odot \iota^t \quad \text{forget and input gates control update of memory} \\
\tilde{s}^t &= s^t \odot o^t \quad \text{output gate controls information that can leave the unit}
\end{aligned}
$$

# An LSTM-based Recurrent Network

**Inputs:** A sequence $x_1 \ldots x_n$ where each $x_j \in \mathbb{R}^d$. A label $y \in \{1 \ldots K\}$.

**Computational Graph:**

- $h^{(0)}, s^{(0)}, \tilde{s}^{(0)}$ are set to some inital values.
- For $t = 1 \ldots n$
  - $s^{(t)}, \tilde{s}^{(t)}, h^{(t)} = \text{LSTM}(x^{(t)}, s^{(t-1)}, \tilde{s}^{(t-1)}, h^{(t-1)}; \theta)$
- $l = V^{lh} h^{(n)} + V^{ls} \tilde{s}^{(n)} + \gamma, \;\; q = \text{LS}(l),\; o = -q_y$

# The Computational Graph

# An LSTM-based Recurrent Network for Tagging

**Inputs:** A sequence $x_1 \dots x_n$ where each $x_j \in \mathbb{R}^d$. A sequence $y_1 \dots y_n$ of tags.

**Computational Graph:**

- $h^{(0)}, s^{(0)}, \tilde{s}^{(0)}$ are set to some inital values.
- For $t = 1 \dots n$
  - $s^{(t)}, \tilde{s}^{(t)}, h^{(t)} = \text{LSTM}(x^{(t)}, s^{(t-1)}, \tilde{s}^{(t-1)}, h^{(t-1)}; \theta)$
- For $t = 1 \dots n$
  - $l^t = V \times \text{CONCAT}(h^{(t)}, \tilde{s}^{(t)}) + \gamma, \;\; q^t = \text{LS}(l^t), \; o^t = -q_{y^t}$
- $o = \sum_{t=1}^{n} o^t$

# The Computational Graph

# A bi-directional LSTM (bi-LSTM) for tagging

**Inputs:** A sequence $x_1 \dots x_n$ where each $x_j \in \mathbb{R}^d$. A sequence $y_1 \dots y_n$ of tags.

**Definitions:** $\theta^F$ and $\theta^B$ are parameters of a forward and backward LSTM.

**Computational Graph:**

- $h^{(0)}, s^{(0)}, \tilde{s}^{(0)}, \eta^{(n+1)}, \alpha^{(n+1)}, \tilde{\alpha}^{(n+1)}$ are set to some inital values.
- For $t = 1 \dots n$
  - $s^{(t)}, \tilde{s}^{(t)}, h^{(t)} = \mathsf{LSTM}(x^{(t)}, s^{(t-1)}, \tilde{s}^{(t-1)}, h^{(t-1)}; \theta^F)$
- For $t = n \dots 1$
  - $\alpha^{(t)}, \tilde{\alpha}^{(t)}, \eta^{(t)} = \mathsf{LSTM}(x^{(t)}, \alpha^{(t+1)}, \tilde{\alpha}^{(t+1)}, \eta^{(t+1)}; \theta^B)$
- For $t = 1 \dots n$
  - $l^t = V \times \mathsf{CONCAT}(h^{(t)}, \tilde{s}^{(t)}, \eta^{(t)}, \tilde{\alpha}^t) + \gamma, \quad q^t = \mathsf{LS}(l^t),$
    $o^t = -q_{y^t}$
- $o = \sum_{t=1}^{n} o^t$

# The Computational Graph

# Results on Language Modeling

| Model | Num. Params [billions] | Training Time [hours] | Training Time [CPUs] | Perplexity |
|---|---|---|---|---|
| Interpolated KN 5-gram, 1.1B n-grams (KN) | 1.76 | 3 | 100 | 67.6 |
| Katz 5-gram, 1.1B n-grams | 1.74 | 2 | 100 | 79.9 |
| Stupid Backoff 5-gram (SBO) | 1.13 | 0.4 | 200 | 87.9 |
| Interpolated KN 5-gram, 15M n-grams | 0.03 | 3 | 100 | 243.2 |
| Katz 5-gram, 15M n-grams | 0.03 | 2 | 100 | 127.5 |
| Binary MaxEnt 5-gram (n-gram features) | 1.13 | 1 | 5000 | 115.4 |
| Binary MaxEnt 5-gram (n-gram + skip-1 features) | 1.8 | 1.25 | 5000 | 107.1 |
| Hierarchical Softmax MaxEnt 4-gram (HME) | 6 | 3 | 1 | 101.3 |
| Recurrent NN-256 + MaxEnt 9-gram | 20 | 60 | 24 | 58.3 |
| Recurrent NN-512 + MaxEnt 9-gram | 20 | 120 | 24 | 54.5 |
| Recurrent NN-1024 + MaxEnt 9-gram | 20 | 240 | 24 | 51.3 |

Table 1: Results on the 1B Word Benchmark test set with various types of language models.

- Results from *One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling*, Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants.

# Results on Dependency Parsing

- *Deep Biaffine Attention for Neural Dependency Parsing*, Dozat and Manning.
- Uses a bidirectional LSTM to represent each word
- Uses LSTM representations to predict head for each word in the sentence
- Unlabeled dependency accuracy: 95.75%

# Conclusions

- Recurrent units map input sequences $x_1 \ldots x_n$ to representations $h^1 \ldots h^n$. The vector $h^n$ can be used to predict a label for the entire sentence. Each vector $h^i$ for $i = 1 \ldots n$ can be used to make a prediction for position $i$
- LSTMs are recurrent units that make use of more involved recurrent updates. They maintain a "memory" state. Empirically they perform extremely well
- Bi-directional LSTMs allow representation of both the information before and after a position $i$ in the sentence
- Many applications: language modeling, tagging, parsing, speech recognition, we will soon see machine translation