# CSE 5523: Lecture Notes 16
## Recurrent Neural Networks

## Contents

16.1 Simple Recurrent Networks [Elman, 1991] . . . . . . . . . . . . . . . . . . . . . . . . 1
16.2 Long Short-Term Memory [Hochreiter and Schmidhuber, 1997] . . . . . . . . . . . 1

Neural networks can be defined to predict hidden states of sequential input.

These networks are recursive, and of potentially unbounded depth, so they re-use models.

### 16.1 Simple Recurrent Networks [Elman, 1991]

A simple recurrent network defines a hidden state vector $\mathbf{h}_t$ at each time step $t$:

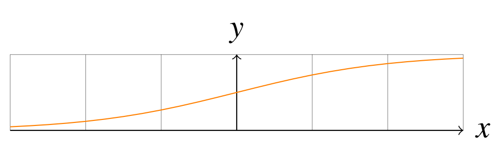
$$\mathbf{h}_t \stackrel{\text{def}}{=} \text{logistic}\left(\mathbf{W}_H \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix}\right)$$

and defines an output vector $\mathbf{y}_t$ based on its hidden state vector:

$$\mathbf{y}_t \stackrel{\text{def}}{=} \text{logistic}(\mathbf{W}_Y\, \mathbf{h}_t)$$

Here logistic is a multinomial logistic function on $\mathbf{x} \in \mathbb{R}^D$ with $D$ units:

$$\text{logistic}(\,\mathbf{x}\,) = \frac{\mathbf{1}}{\mathbf{1} + \exp(\,-\mathbf{x}\,)}$$




Importantly, the weight matrixes $\mathbf{W}_H$ and $\mathbf{W}_Y$ do not depend on the time step.

This is called **stationarity**.

### 16.2 Long Short-Term Memory [Hochreiter and Schmidhuber, 1997]

Recurrent networks can lose information through backprop (vanishing and exploding gradients).

This can be mitigated using ‘cell’ memories which are stored and retrieved using learned ‘gates’.

LSTMs maintain a hidden state $\mathbf{h}_t$ and a memory cell $\mathbf{c}_t$ at each time step $t$.

The cell retains input, if it's judged to be important:

$$\mathbf{c}_t \stackrel{\text{def}}{=} \underbrace{\tanh\left(\mathbf{G}\begin{bmatrix}\mathbf{h}_{t-1}\\ \mathbf{x}_t\end{bmatrix}\right)}_{\text{gate to reformat input}} \odot \underbrace{\text{logistic}\left(\mathbf{I}\begin{bmatrix}\mathbf{h}_{t-1}\\ \mathbf{x}_t\end{bmatrix}\right)}_{\text{gate to store new content}} + \underbrace{\text{logistic}\left(\mathbf{F}\begin{bmatrix}\mathbf{h}_{t-1}\\ \mathbf{x}_t\end{bmatrix}\right)}_{\text{gate to forget old content}} \odot \mathbf{c}_{t-1}$$

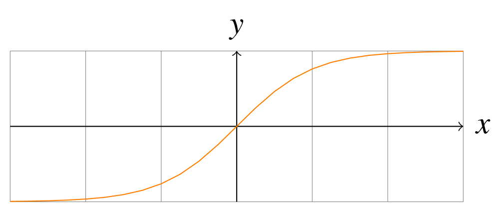
The hidden state controls the output (which may be connected to a higher layer):

$$\mathbf{h}_t \stackrel{\text{def}}{=} \tanh(\mathbf{c}_t) \odot \underbrace{\text{logistic}\left(\mathbf{O}\begin{bmatrix}\mathbf{h}_{t-1}\\ \mathbf{x}_t\end{bmatrix}\right)}_{\text{gate to output from cell}}$$

Here tanh is a hyperbolic tangent function, also on $\mathbf{x} \in \mathbb{R}^D$ with $D$ units:

$$\begin{aligned}
\tanh(\mathbf{x}) &= \frac{\exp(\mathbf{x}) - \exp(-\mathbf{x})}{\exp(\mathbf{x}) + \exp(-\mathbf{x})} & \\
&= \frac{\exp(\mathbf{x}) + \exp(\mathbf{x}) - \exp(\mathbf{x}) - \exp(-\mathbf{x})}{\exp(\mathbf{x}) + \exp(-\mathbf{x})} & \text{add } \exp(\mathbf{x}) - \exp(\mathbf{x}) \\
&= 2\frac{\exp(\mathbf{x})}{\exp(\mathbf{x}) + \exp(-\mathbf{x})} - \mathbf{1} & \text{multiplicative inverse} \\
&= 2\frac{\exp(2\mathbf{x})}{\exp(2\mathbf{x}) + \mathbf{1}} - \mathbf{1} & \text{multiply first term by } \exp(\mathbf{x}) \\
&= 2\,\text{logistic}(2\mathbf{x}) - \mathbf{1} & \text{definition of logistic}
\end{aligned}$$

(it's essentially just a re-scaled logistic).




We can efficiently apply the gates by stacking them up $\begin{bmatrix}\mathbf{F}\\ \mathbf{G}\\ \mathbf{I}\\ \mathbf{O}\end{bmatrix}$ before multiplying with $\begin{bmatrix}\mathbf{h}_{t-1}\\ \mathbf{x}\end{bmatrix}$.

# References

[Elman, 1991] Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7:195–225.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.