Neural Language Models

Neural Unit

$$z = \mathbf{w} \cdot \mathbf{x} + b$$
$$y = f(z)$$

w = weight vector, x = input, b = bias scalar, f = activation function, y = output of unit

Activation Function

$z = \max(0, k)$

Others: tanh, sigmoid, leaky ReLU

Neural Network

$$\mathbf{h} = f(\mathbf{W}\mathbf{x} + \mathbf{b})$$

h = hidden layer vector, f = activation function, W = Weights matrix, x = input vector, b = bias vector

Neural Network

Here are the final equations for a feedforward network with a single hidden layer, which takes an input vector $\mathbf{x}$, outputs a probability distribution $\mathbf{y}$, and is parameterized by weight matrices $\mathbf{W}$ and $\mathbf{U}$ and a bias vector $\mathbf{b}$:

$$\begin{aligned} \mathbf{h} &= \sigma(\mathbf{W}\mathbf{x}+\mathbf{b}) \\ \mathbf{z} &= \mathbf{U}\mathbf{h} \\ \mathbf{y} &= \text{softmax}(\mathbf{z}) \end{aligned} \tag{7.12}$$

OOV? Embeddings

- encode semantic information
- can be pretrained (BERT, fasttext etc)

Feedforward Neural Networks

- information is passed forward
- fixed context size
- each unit in layer i is connected to each unit in layer i+1, and there are no cycles.
- use embeddings

Recurrent Neural Networks

$$h_t = g(Uh_{t-1} + Wx_t)$$

w = weight vector, x = input, b = bias scalar, f = activation function, y = output of unit

Recurrent Neural Networks

- Stacked RNNs: differing layers of abstractions captured between each RNN
- Bidirectional RNNs: two RNNs, left-to-right and right-to-left

Recurrent Neural Networks

- context is no longer a limit
- vanishing gradient, difficult to store distant memory

Long Short-Term Memory

- when to forget and when to remember?
- specialized context layer
- forget, add, output gates

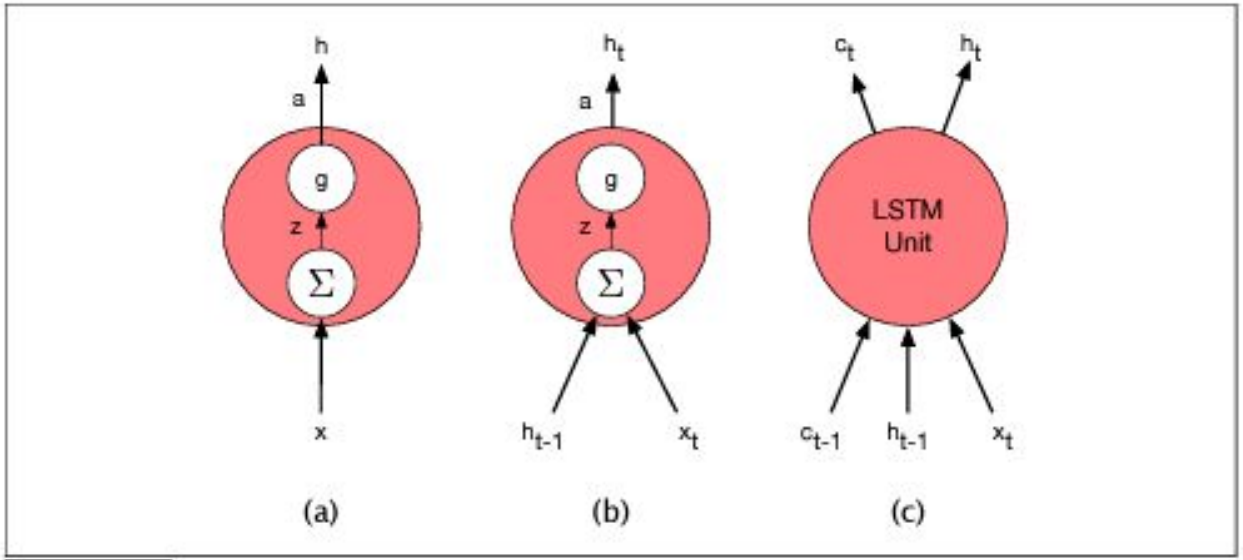

Figure 9.14 Basic neural units used in feedforward, simple recurrent networks (SRN), and long short-term memory (LSTM).

Seq2Seq

- 1) Encode the input sequence into state vectors.
- 2) Start with a target sequence of size 1 (just the start-of-sequence character).
- 3) Feed the state vectors and 1-char target sequence to the decoder to produce predictions for the next character.
- 4) Sample the next character using these predictions (we simply use argmax).
- 5) Append the sampled character to the target sequence
- 6) Repeat until we generate the end-of-sequence character or we hit the character limit.