TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2020

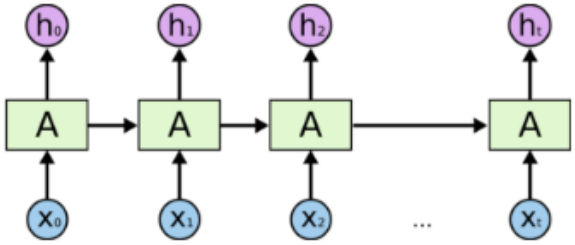
Recurrent Neural Networks (RNNs)

Word Embeddings

Each word w is associated with a vector $e(w)[I]$ called the embedding of word w.

The matrix e can be viewed as a dictionary assigning each word w the vector $e(w)[I]$.

Recurrent Neural Network (RNN) Language Modeling

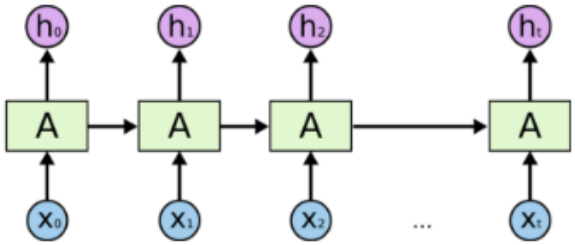


[Christopher Olah]

A typical RNN neural language model has the form

$$P_\Phi(w_t \mid w_0, \cdots, w_{t-1}) = \underset{w_t}{\text{softmax}}\, e(w_t)[I]h[t-1, I]$$

Vanilla RNNs



[Christopher Olah]

A Vanilla RNN uses two-input linear threshold units.

$$h[b,t,j] = \sigma\left(W^{h,h}[j,I]h[b,t-1,I] + W^{x,h}[j,K]x[b,t,K] - B[j]\right)$$

Exploding and Vanishing Gradients

If we avoid saturation of the activation functions then we get exponentially growing or shrinking eigenvectors of the weight matrix.

Note that if the forward values are bounded by sigmoids or tanh then they cannot explode.

However the gradients can still explode.

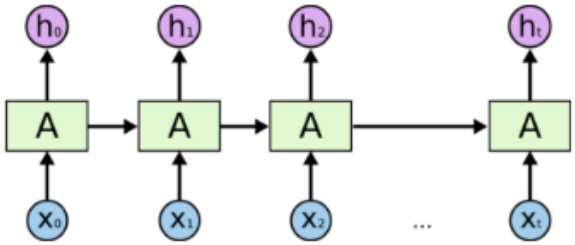
Exploding Gradients: Gradient Clipping

We can dampen the effect of exploding gradients by clipping them before applying SGD.

$$W.\text{grad}' = \begin{cases} W.\text{grad} & \text{if } ||W.\text{grad}|| \leq n_{\max} \\ \\ n_{\max}\, W.\text{grad}/||W.\text{grad}|| & \text{otherwise} \end{cases}$$

See `torch.nn.utils.clip_grad_norm`

Time as Depth



[Christopher Olah]

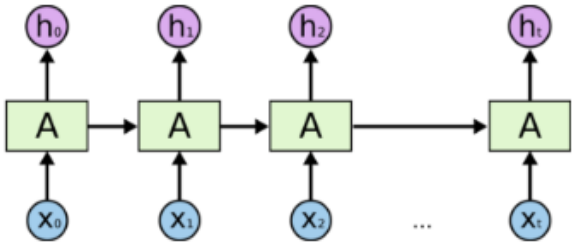
We would like the RNN to remember and use information from much earlier inputs.

All the issues with depth now occur through time.

However, for RNNs at each time step we use the same model parameters.

In CNNs at each layer uses its own model parameters.

"Residual Connections" Through Time

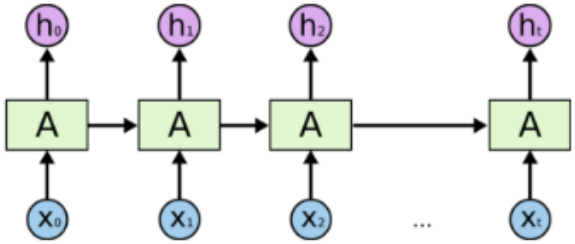


[Christopher Olah]

We would like to have residual connections through time.

However, we have to handle the fact that the same model parameters are used at every time step.

Gated RNNs



[Christopher Olah]

$$h[b,t,j] = G_t[b,t,j]h[b,t-1,j] + (1 - G[b,t,j])R[b,t,j]$$

This is analogous to a residual connection.

Rather than add the "next layer" $R[b,t,j]$ to the input $h[b,t-1,j]$ as in a residual connection, we take a convex combination determined by a computed "gate" $G[b,t,j] \in [0,1]$.

Update Gate RNN (UGRNN)

$$R[b,t,j] = \tanh\left(W^{h,R}[j,I]h[b,t-1,I] + W^{x,R}[j,K]x[b,t,K] - B^R[j]\right)$$

$$G[b,t,j] = \sigma\left(W^{h,G}[j,I]h[b,t-1,I] + W^{x,G}[j,K]x[b,t,K] - B^R[j]\right)$$

$$h[b,t,j] = G[b,t,j]h[b,t-1,j] + (1 - G[b,t,j])R[b,t,j]$$

$$\Phi = (W^{h,R}, W^{x,R}, B^R, W^{h,G}, W^{x,G}, B^G)$$

$$\tanh(x) \in (-1,1) \qquad \sigma(x) \in (0,1)$$

Hadamard product

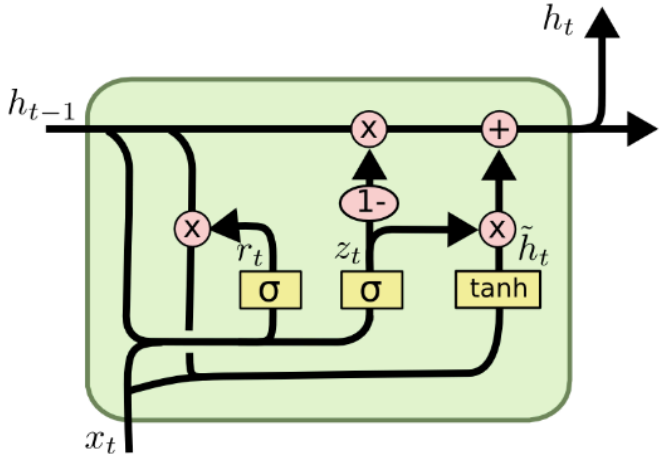
$$h[b,t,j] = G[b,t,j]h[b,t-1,j] + (1 - G[b,t,j])R[b,t,j]$$

is sometimes written as

$$h[b,t,J] = G[b,t,J] \odot h[b,t-1,J] + (1 - G[b,t,J]) \odot R[b,t,J]$$

$\odot$ is the Hadamard product (componentwise product) on vectors.

Gated Recurrent Unity (GRU) by Cho et al. 2014

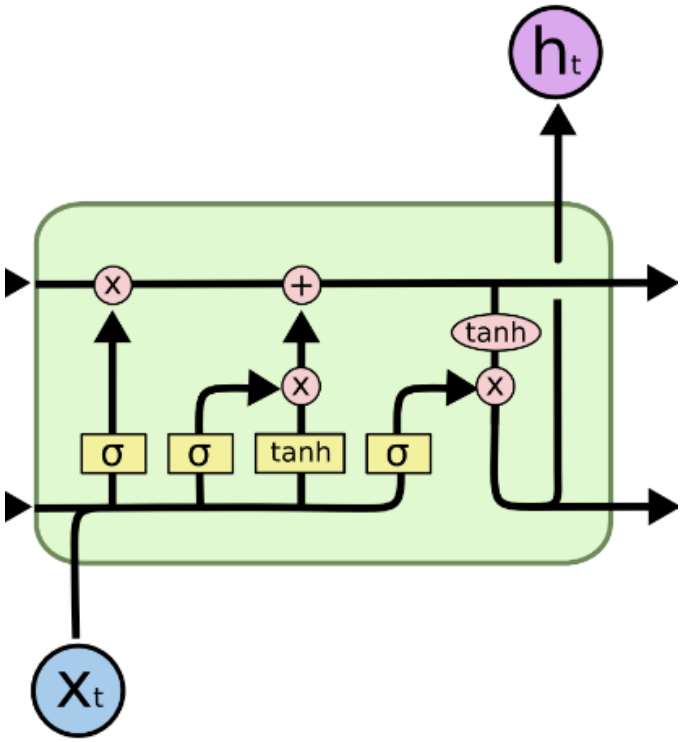


[Christopher Olah]

The right half is a UGRNN.

The GRU adds a gating on h_{t-1} before the tanh.

Long Short Term Memory (LSTM)



[figure: Christopher Olah]

[LSTM: Hochreiter&Shmidhuber, 1997]

UGRNN vs. GRUs vs. LSTMs

In class projects from previous years, GRUs consistently outperformed LSTMs.

A systematic study [Collins, Dickstein and Sussulo 2016] states:

> Our results point to the GRU as being the most learnable of gated RNNs for shallow architectures, followed by the UGRNN.

RNN for a generic CELL Procedure

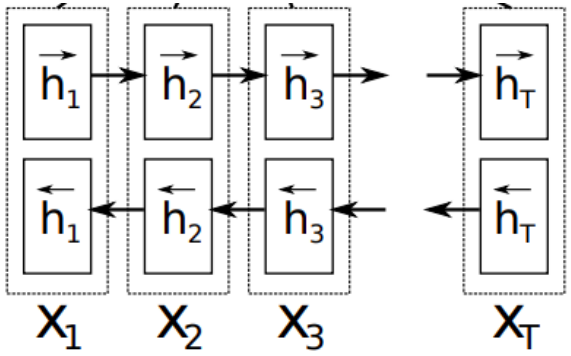
As usual, we use capital letter indeces to denote whole tensors or slices and lower case letters to denote particular index values.

Procedure $\mathrm{RNN}_{\Phi}(x(T, I))$

$$\begin{aligned} h[0, J] &= \mathrm{CELL}_{\Phi.\mathrm{cell}}(\Phi.\mathrm{init}[J],\ x[0, I]) \\ \text{for } t > 0 \quad h[t, J] &= \mathrm{CELL}_{\Phi.\mathrm{cell}}(h[t-1, J],\ x[t, I]) \end{aligned}$$

Return $h[T, J]$

bidirectional RNNS

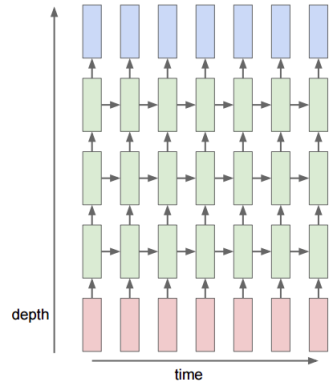


$$\vec{h}[T, J] = \vec{\mathrm{RNN}}_{\Phi.LR}(x[T, I])$$

$$\overleftarrow{h}[T, J] = \overleftarrow{\mathrm{RNN}}_{\Phi.RL}(x[T, I])$$

$$h[t, 2J] = \vec{h}[t, J]; \overleftarrow{h}[t, J]$$

Multi-Layer RNNs



We let ℓ indicate the layer of a multi-layer RNN.

$$h[0, T, J] = \text{RNN}_{\Phi[0]}(x[T, I])$$
$$\text{for } \ell > 0 \quad h[\ell, T, J] = \text{RNN}_{\Phi[\ell]}(h[\ell - 1, T, J])$$

Each layer can be bidirectional.

Residual Multi-Layer RNNs

But layers are now typically stacked using residual connections.

$$h[0, T, J] = \mathrm{RNN}_{\Phi[0]}(x[T, I])$$

$$\text{for } \ell > 0 \quad h[\ell, T, J] = h[\ell - 1, T, J] + \mathrm{RNN}_{\Phi[\ell]}(h[\ell - 1, T, J])$$

END