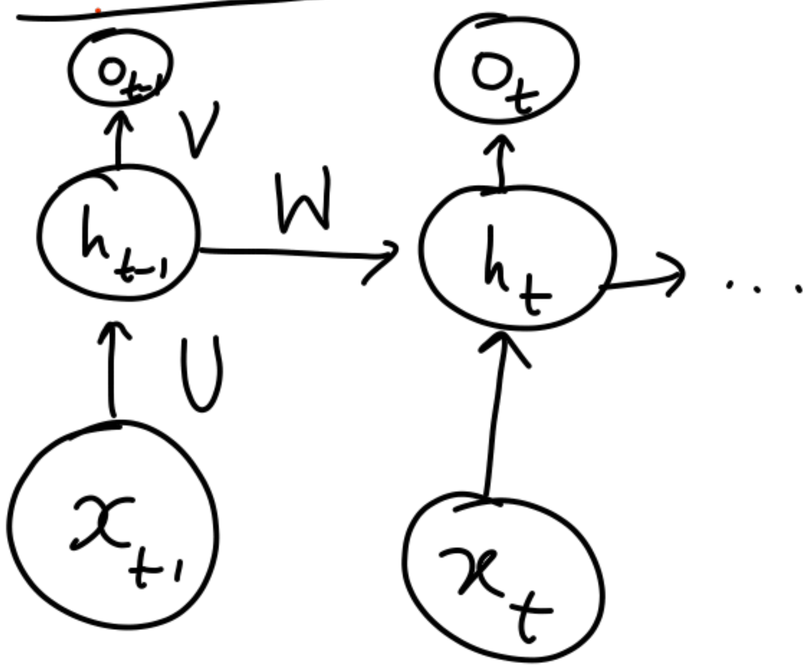
CS725



Recurrent state

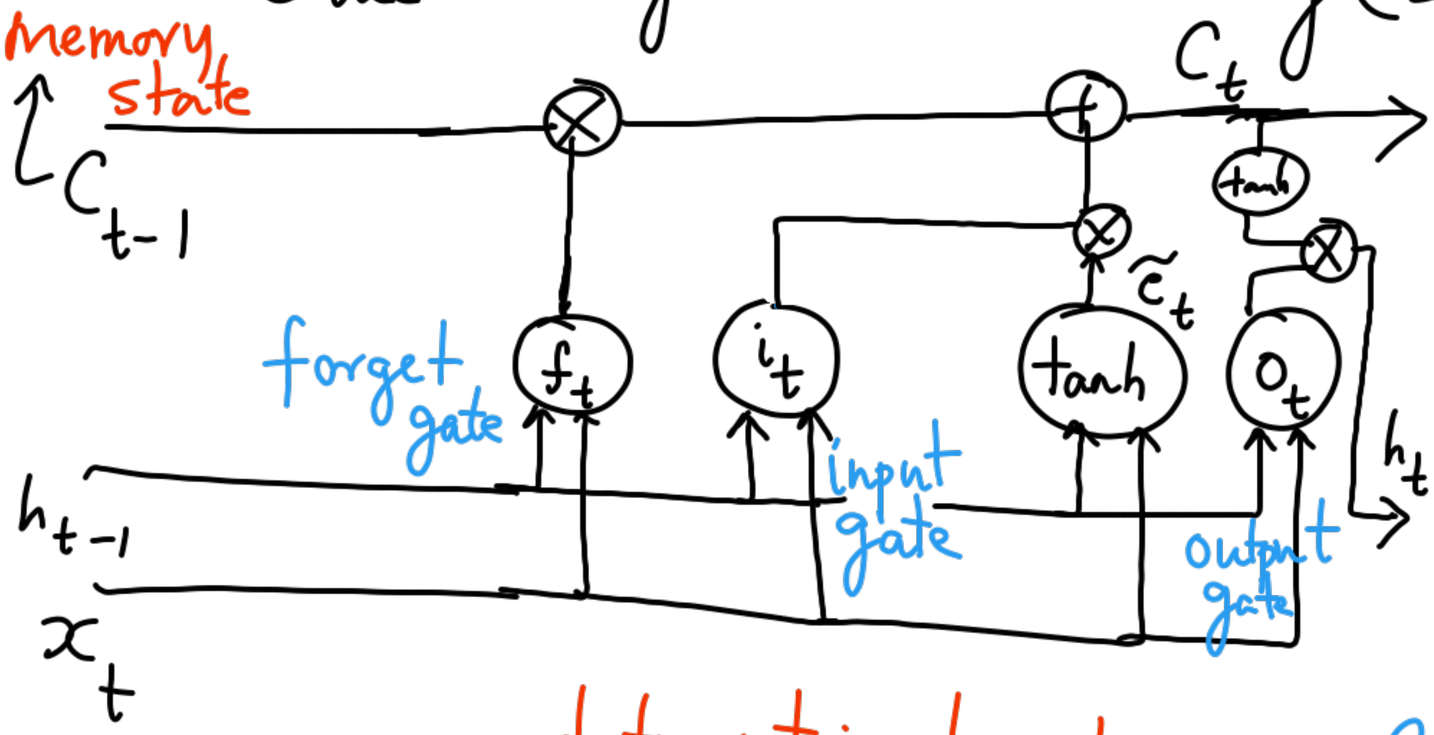
$$h_t = \tanh(Ux_t + Wh_{t-1} + b)$$

Problems of vanishing / exploding gradients

How do we handle this?

Could be handled by clipping threshold

One approach to handle vanishing gradients is a new type of RNN called "Long short-term memory (LSTM)" units



$$f_t = \sigma\left(W^f \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} + b^f\right)$$

$$i_t = \sigma\left(W^i \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} + b^i\right)$$

$$\tilde{C}_t = \tanh\left(W^c \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} + b^c\right)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{c}_t$$

$$o_t = \sigma\left(W^o \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} + b^o\right); \quad h_t = o_t \odot \tanh(C_t)$$

memory state retains long-term information

Training RNNs: Backpropagation through time (BPTT)

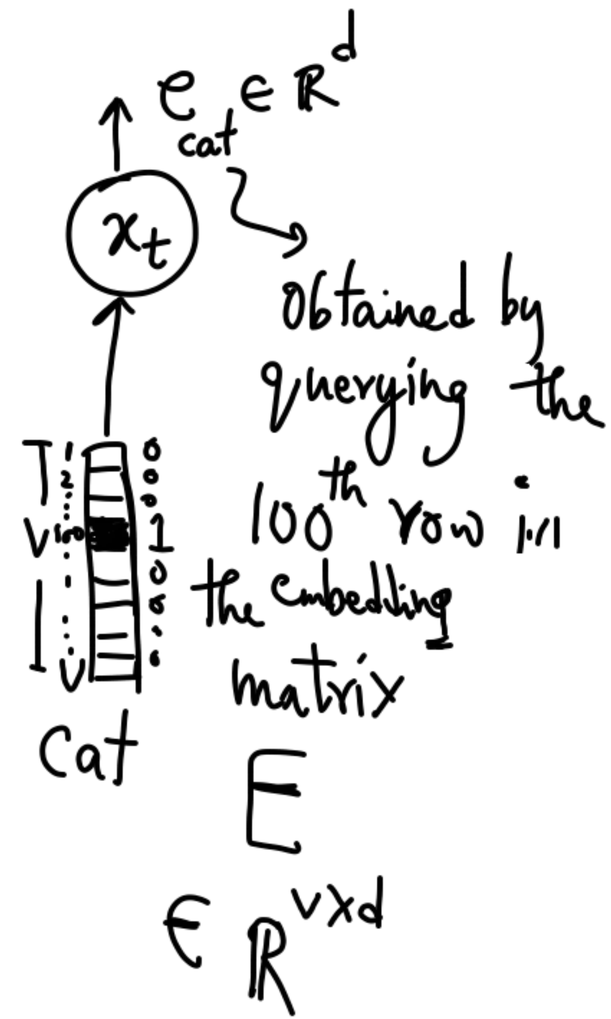
1. Unroll the RNN for as many timesteps as in the input
2. Compute gradients for each weight
3. Average (or sum) the gradients across all connections that should have the same wt
4. Do weight update with these averaged gradients

s.t. all tied shared wts stay the same

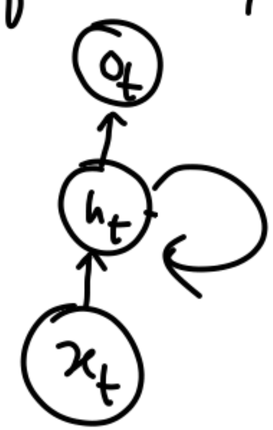
Two points:

(A) Word embeddings:

Maps from a sparse one-hot vector of size V (vocab size) to a dense representation referred to as a "word embedding" of size d, $d \ll V$



(B)



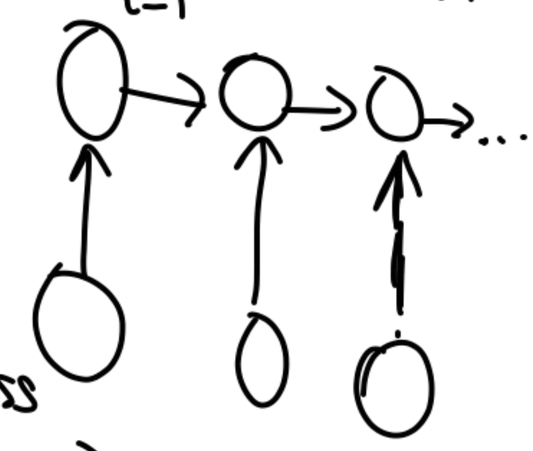
If $o_t \in \mathbb{R}^d$, add a linear layer $W \in \mathbb{R}^{d \times V}$ followed by a softmax

Encoder-Decoder Models

Recall the many-to-many setting, e.g. translation, ASR, etc.

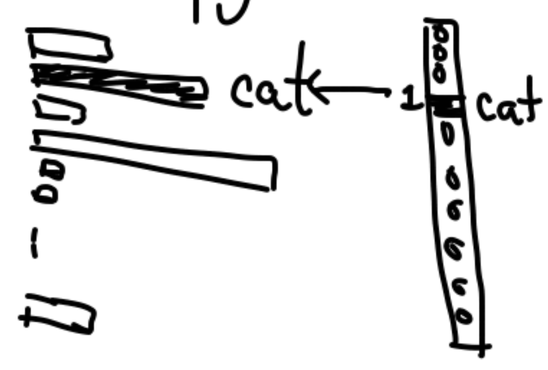
How do we use an RNN for such problems?

$$P(y_1, \dots, y_n) = \prod_{t=1}^{n} P(y_t \mid y_{1:t-1})$$



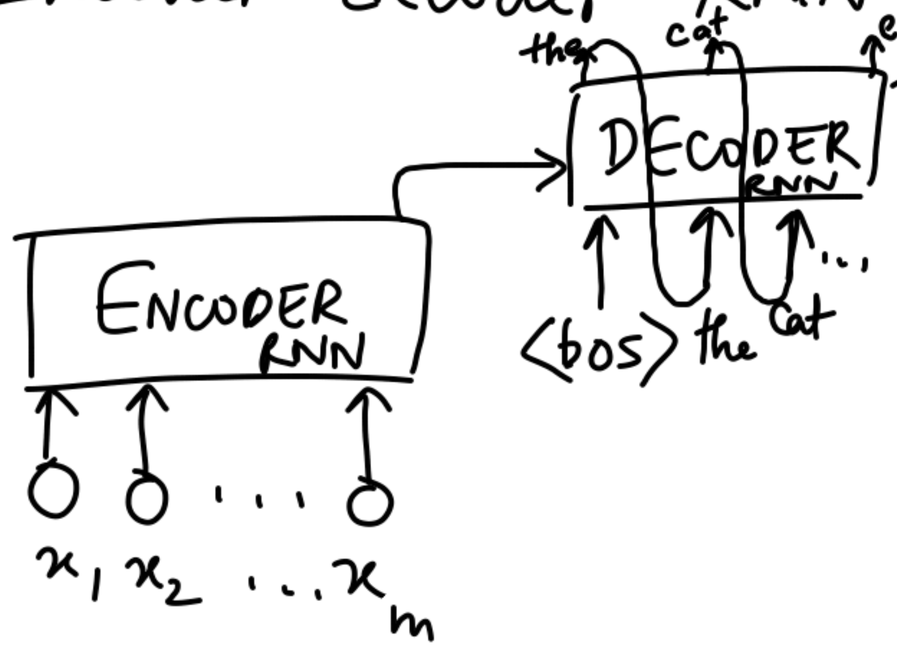
Detour: Train a language model RNN using cross-entropy loss.

At time step t, softmax prediction $P(y_t \mid y_{<t})$ $\Rightarrow$



$\Rightarrow$ Cross-entropy loss $-\log P(\text{cat} \mid y_{<t})$

Encoder-decoder RNN for problems like MT, ASR, etc.



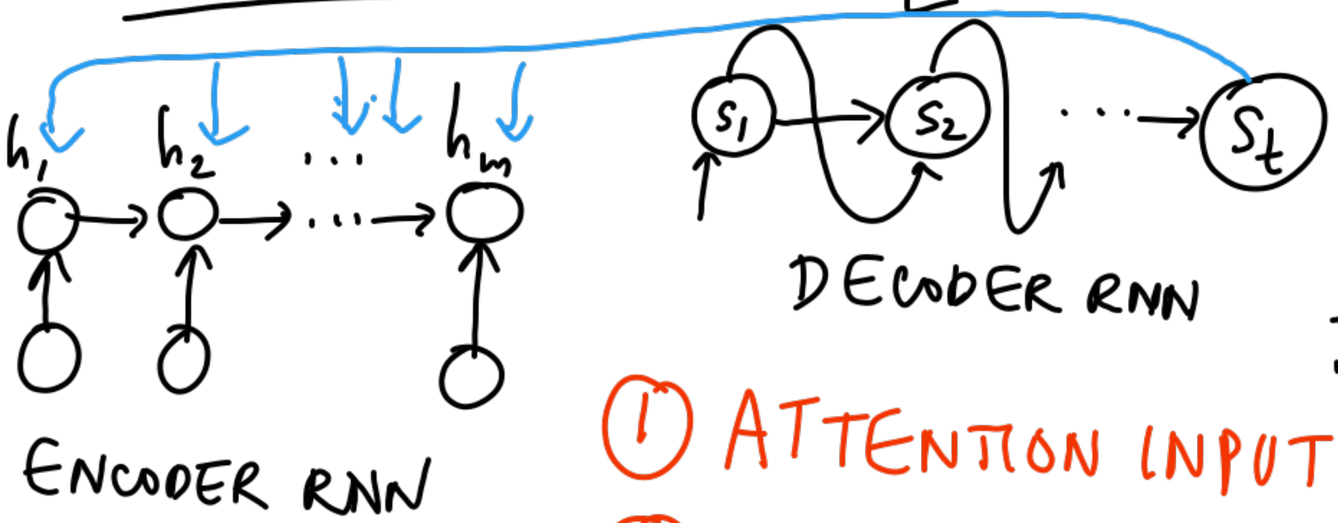
→ Encoder RNN "summarizes" the input into a representation

→ Decoder RNN starts with the summary representation from the encoder and unrolls till it produces end-of-sentence.

Challenges/Limitations:

1. Summary representation may be limited in expressivity esp when m is large
2. Decoder RNN might want to focus on diff aspects of the input at diff time steps

ATTENTION [addresses limitation ② of encoder decoder models]



Attention from s_t over $h_1, \ldots, h_m$

Attention allows decoder states to focus on specific parts of the input

① ATTENTION INPUT: $h_1, \ldots, h_m$ and s_t

② ATTENTION SCORE: $score(h_j, s_t)$ (captures relnship b/w s_t and h_j)

e.g. $score(h_j, s_t) = h_j^T s_t$

③ ATTENTION PROBABILITIES:

$$\alpha_{jt} = \frac{\exp(score(h_j, s_t))}{\sum_j \exp(score(h_j, s_t))}$$

④ ATTENTION OUTPUT:

$$c_t = \sum_j \alpha_{jt} h_j$$