Recurrent Neural Networks

Gautam Kamath

Sequence Models

- Stock prices: given past stock prices $x_1, \dots, x_n$, predict $\hat{x}_t \sim \Pr(x_t | x_1, \dots, x_{t-1})$
- Language modelling: Given phrase, how likely is it?
  - $\Pr[\text{The cat is black}]$ versus $\Pr[\text{is black cat the}]$
  - Note $\Pr[\text{The cat is black}] = \Pr[\text{The}] \cdot \Pr[\text{cat}|\text{The}] \cdot \Pr[\text{is}|\text{The cat}] \cdot \Pr[\text{black}|\text{The cat is}]$
  - Next word prediction: predict $\hat{x}_t \sim \Pr(x_t | x_1, \dots, x_{t-1})$

Idea: Summarize past observations with "state"

- Summary h_t (vector). Then $\hat{x}_t \sim \Pr[x_t|h_t]$.
- h_t is some function of $x_1, \ldots, x_{t-1}$
- Let $h_1 = g(\vec{0}, x_1)$, where g is some function. Then $\hat{y}_1 \sim \Pr[y_1|h_1]$
  - For next word prediction, think $y_1 = x_2$
- Let $h_2 = g(h_1, x_2)$. Then $\hat{y}_2 \sim \Pr[y_2|h_2]$
- (Draw in diagram form: $x_{t-1} \rightarrow h_{t-1} \rightarrow \hat{y}_{t-1}$ and same for t)
- But how do we determine the functions on each edge here?
- How do we compute hidden state?

Recurrent Neural Networks

- $h_t = f(h_{t-1}, x_t, \theta)$
  - Function of previous hidden state, current input, parameter vector
  - First two are time indexed, but parameter vector is fixed
    - A type of "parameter sharing" (like CNNs)
  - Choose f to be a neural network
- (Draw: $x \rightarrow_U h$, self-loop with box on h with parameter W, then draw unrolled)
  - $h_t = \tanh(Wh_{t-1} + Ux_t + b)$
- (Add on $h \rightarrow_V \hat{y} \rightarrow \ell \leftarrow y$)
  - $\hat{y}_t = \text{softmax}(Vh_t + c)$, $\ell\left(\{x_1, \dots, x_n\}, \{y_1, \dots, y_n\}\right) = \sum_{t=1}^{n} \ell(\hat{y}_t, y_t)$

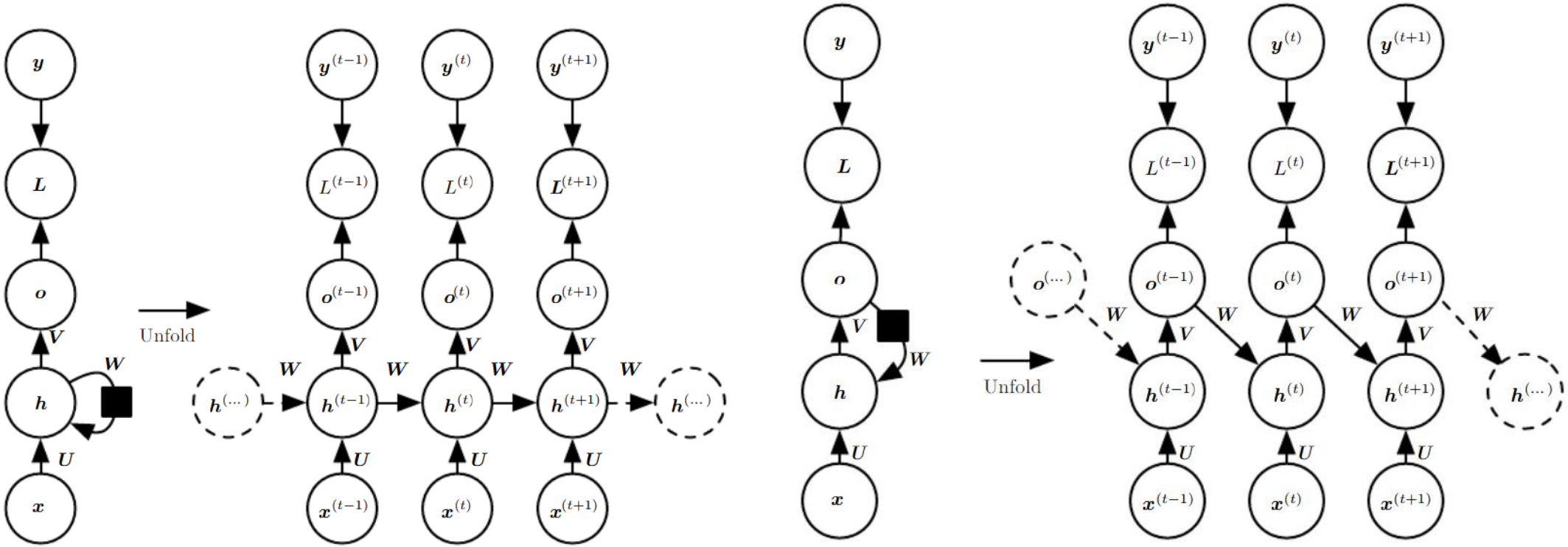
Example: Next Character Prediction

- (Draw: next char prediction. Input is prefix of “hello” (one-hot encoding), choose some hidden layer values (3-dim), and output predictions (4-dim), mark them right or wrong for predictions “ello”)
  - (Draw: lines showing at test time you feed it back into itself)

Other types of sequence problems

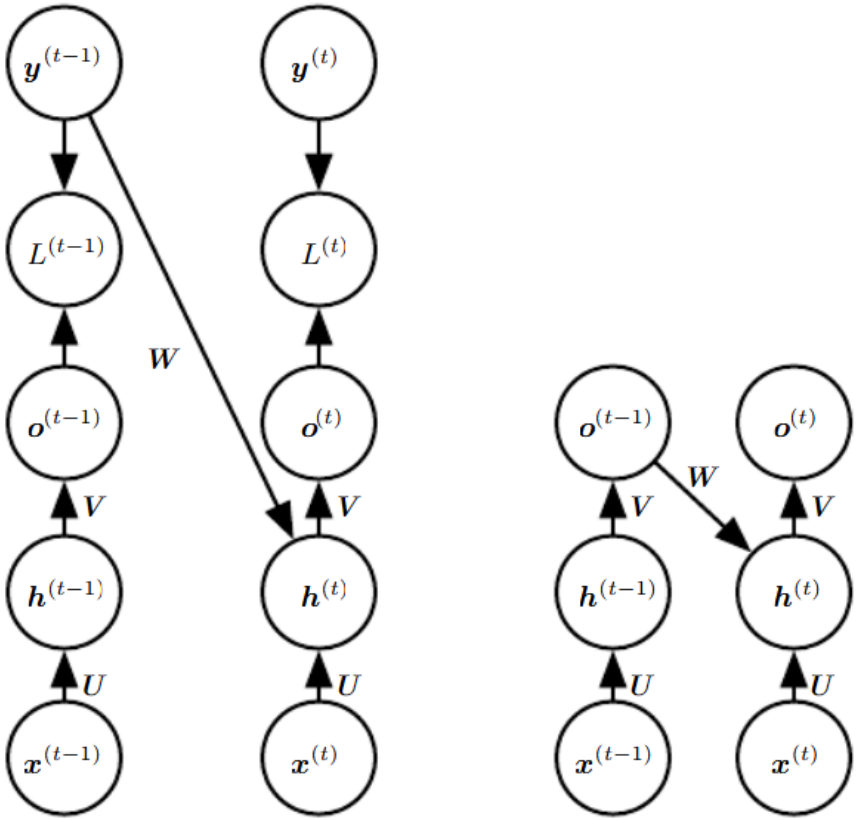
- Many to one (draw)
  - Sentiment classification
  - “I hated this movie” label = -1, “This movie was great!” label = +1
- One to many (draw)
  - Image captioning
- Many to many (draw)
  - Machine translation, “the cat is black” to “le chat est noir”
  - May need to see whole input before translating: “对不起” to “sorry”

Sequential versus parallel



Sequential structure is more powerful…

But parallel enables teacher forcing



Optimization for RNNs

- “Backpropagation through time”
- Vanishing or exploding gradients
  - Gradient magnitude goes to 0 or becomes very large, a symptom of gradients through many layers
- Truncate gradient chains after some τ steps
- Gradient clipping (draw)
  - If $\|g\|_2 \geq v$, then $g \leftarrow (g/\|g\|_2)v$

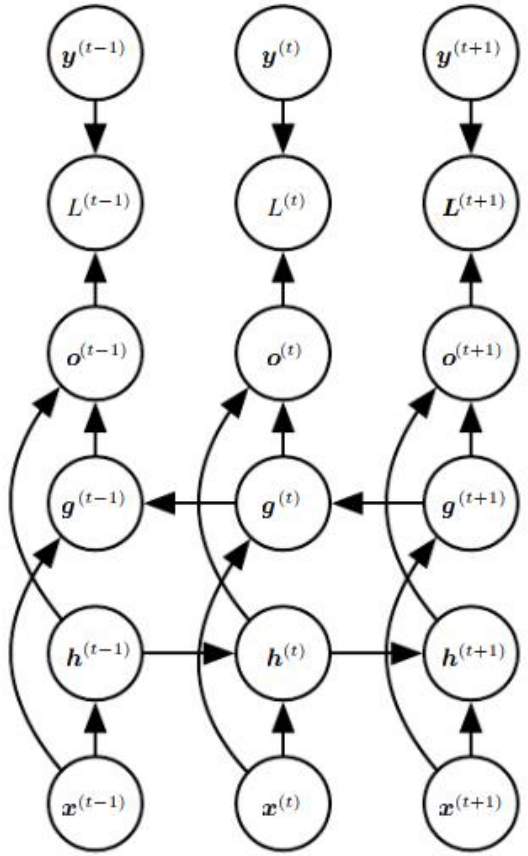
Gated Recurrent Unit (GRU) (Draw as we go)

- Three reasonable things to do with hidden state h_t at a given time:
  1. Let it be a normal update ($h_t = \tanh(Ux_t + Wh_{t-1} + b)$)
  2. Drop the past hidden state and start again ($h_t = \tanh(Ux_t + b)$)
  3. Just use the previous hidden state ($h_t = h_{t-1}$)
- Define $R_t = \sigma(U^{(r)}x_t + W^{(r)}h_{t-1} + b^{(r)})$, $Z_t = \sigma(U^{(z)}x_t + W^{(z)}h_{t-1} + b^{(z)})$
  - Both R_t (reset) and Z_t (update) will be same dimension at h_t, all entries in $[0,1]$
- Let $\tilde{h}_t = \tanh(Ux_t + W(R_t \odot h_{t-1}) + b)$
  - $\odot$ is an entry-wise product. If reset = 0, then we ignore past hidden state (2.), otherwise regular update (1.)
- $h_t = Z_t \odot h_{t-1} + (1 - Z_t) \odot \tilde{h}_t$
  - If update = 1, then we use previous hidden state (3.). Otherwise, use proposal.
- Not binary: actually do some linear combination of these
- More complex structures (Long short-term memory, LSTM)

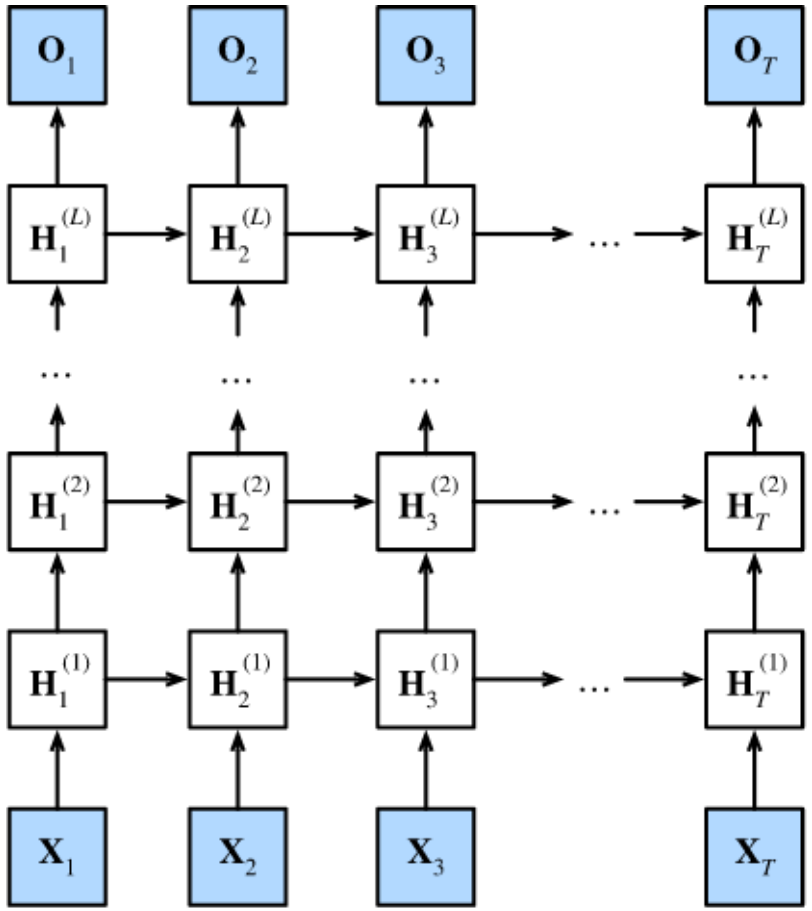
Bidirectional RNNs

I went to the bank…

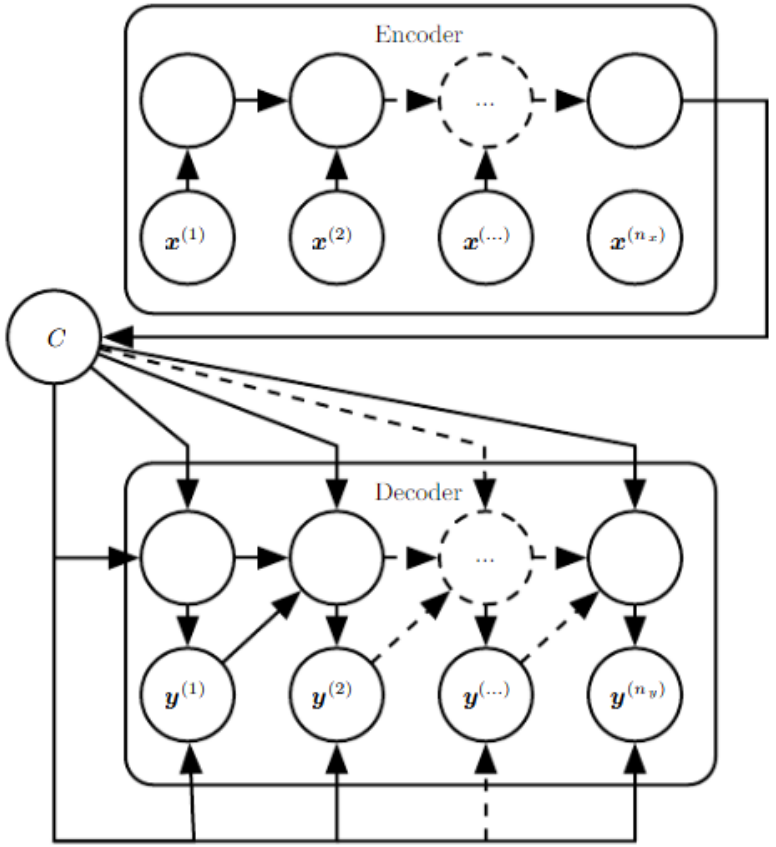
1. Of the river
2. To withdraw money



Deep RNNs


$\mathbf{O}_1$
$\mathbf{O}_2$
$\mathbf{O}_3$
$\mathbf{O}_T$
$\mathbf{H}_1^{(L)}$
$\mathbf{H}_2^{(L)}$
$\mathbf{H}_3^{(L)}$
$\mathbf{H}_T^{(L)}$
$\mathbf{H}_1^{(2)}$
$\mathbf{H}_2^{(2)}$
$\mathbf{H}_3^{(2)}$
$\mathbf{H}_T^{(2)}$
$\mathbf{H}_1^{(1)}$
$\mathbf{H}_2^{(1)}$
$\mathbf{H}_3^{(1)}$
$\mathbf{H}_T^{(1)}$
$\mathbf{X}_1$
$\mathbf{X}_2$
$\mathbf{X}_3$
$\mathbf{X}_T$

Encoder-Decoder Architecture



E.g., machine translation