Recurrent Neural Networks

Sequence models

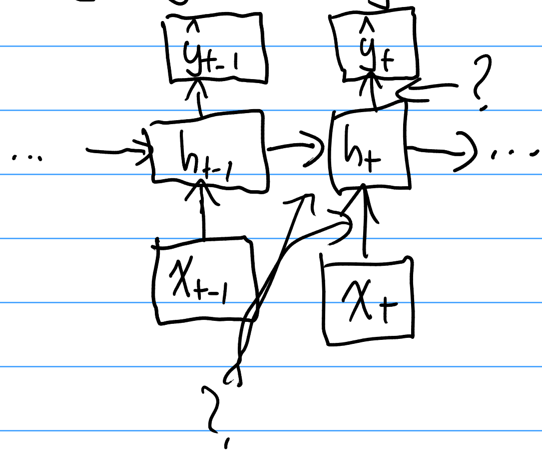
Examples:

- Stock prices: $x_1, \dots, x_n$
  Predict $\hat{x}_t \sim \Pr(x_t | x_1, \dots, x_{t-1})$
- Language models:
  Given phrase, what is its prob?
  $\Pr[\text{The cat is black}] = \Pr[\text{The}] \cdot \Pr[\text{cat}|\text{The}] \cdot \Pr[\text{is}|\text{The, cat}] \cdot \Pr[\text{black}|\text{The, cat, is}]$

  Next word predict:
  $\hat{x}_t \sim \Pr[x_t | x_1, \dots, x_{t-1}]$
  next word — prev words

Idea: Summarize past observations.
Summary h_t (vector). Then $\hat{x}_t \sim \Pr[x_t | h_t]$
h_t: fn of $x_1, \dots, x_t$. Update over time.
$h_1 = g(\vec{0}, x_1)$. $h_2 = g(h_1, x_2)$ . . .
$\hat{y}_1 \sim \Pr[y_1|h_1]$ $\hat{y}_2 \sim \Pr[y_2|h_2]$

(e.g. $y_1 = x_2$)
Next word prediction



How to compute hidden state?

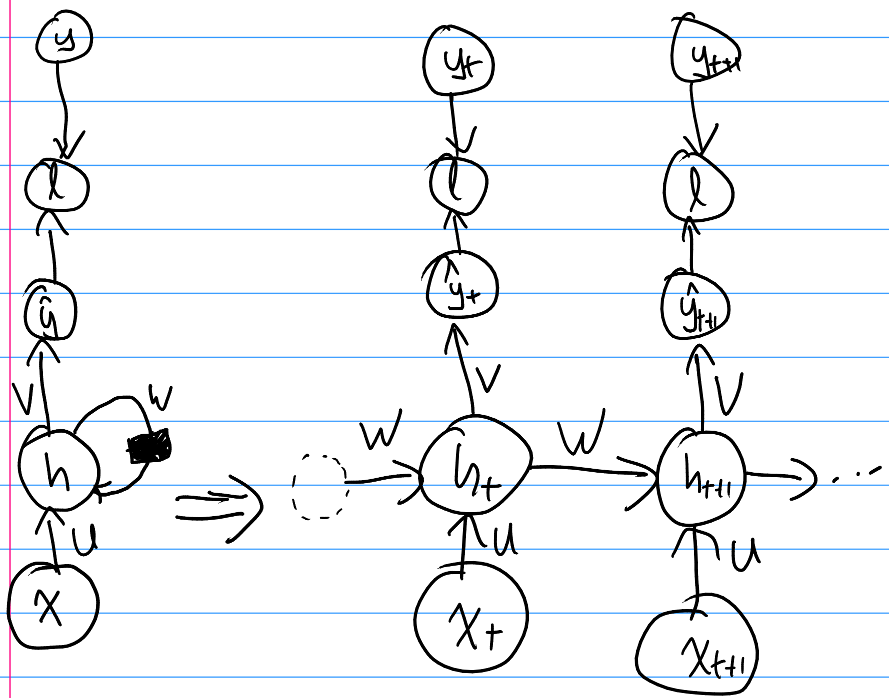
$$h_t = f(h_{t-1}, x_t, \theta)$$

NN

prev hidden state

current input

parameter vector

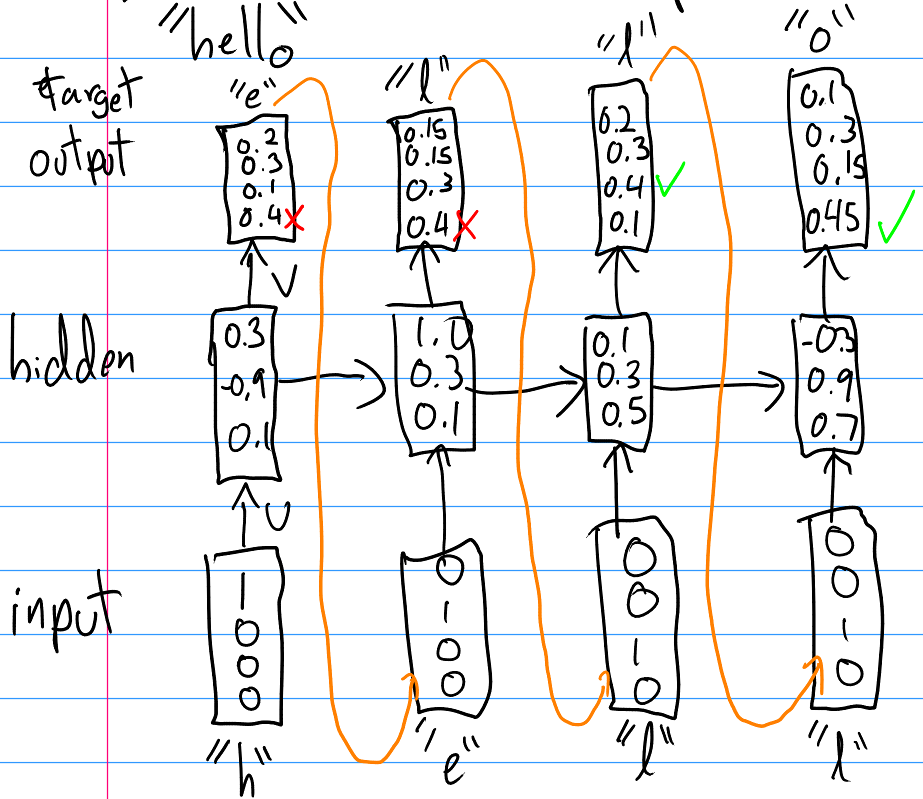


$$h_t = \tanh(Wh_{t-1} + Ux_t + b)$$

$$\hat{y}_t = \mathrm{softmax}(Vh_t + c)$$

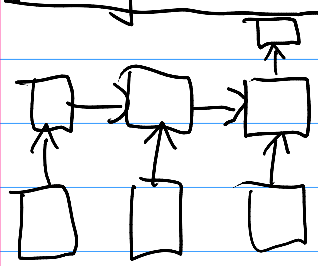
$$l(\{x_1, .., x_n\}, \{y_1, .., y_n\}) = \sum_{t=1}^{n} l(y_t, \hat{y}_t)$$

Ex: Next char prediction



Ex. of oth seq. problems:

Many to one



Sentiment classification

"this movie is great" → +1

"I hated this movie" → -1

One to many

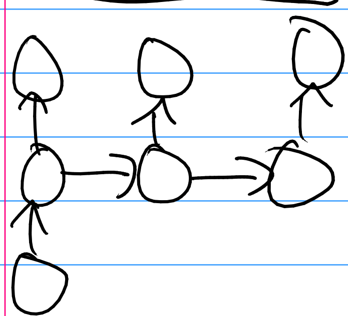
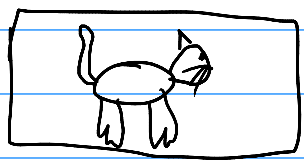
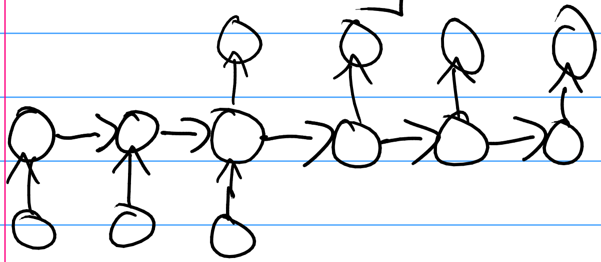


Image caption



A very poorly drawn cat

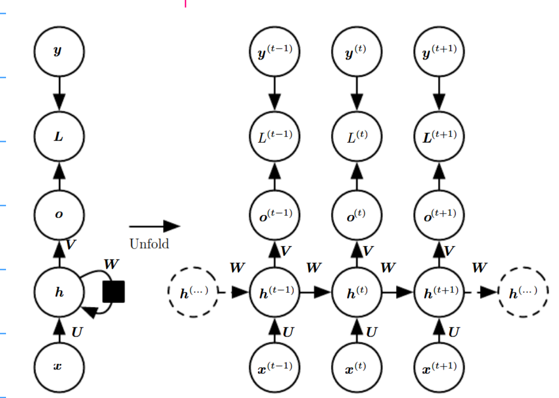
Many to many

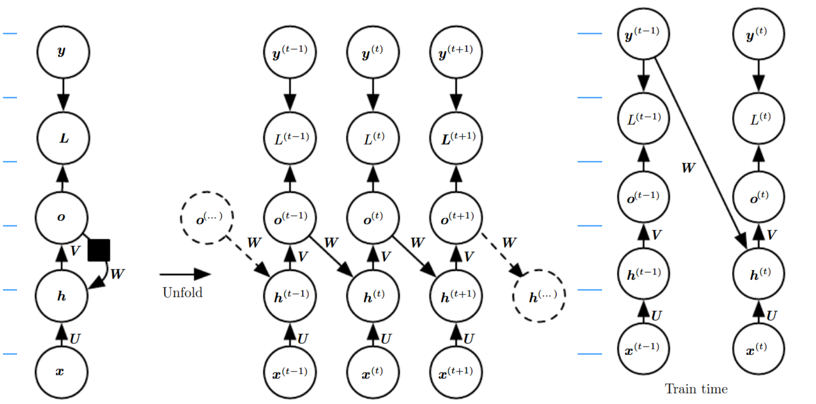


Machine translation
The cat is black
⇒ Le chat est noir

对不起 → Sorry

Sequential vs Parallel





Teacher forcing

More powerful

Parallelizable

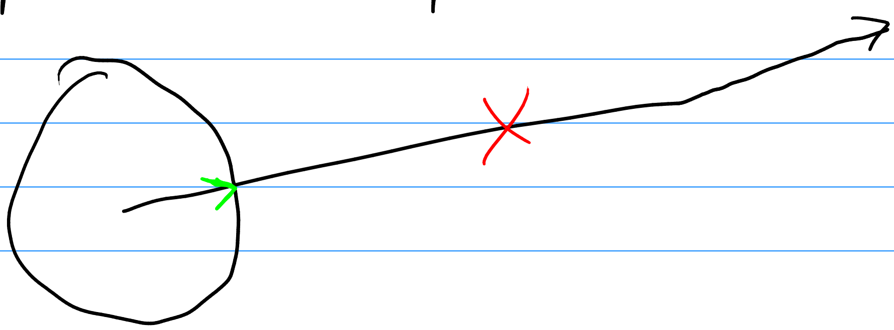
Optimization for RNNs

"Backpropagation through time"

Long chains cause issues
Vanishing or exploding gradients.

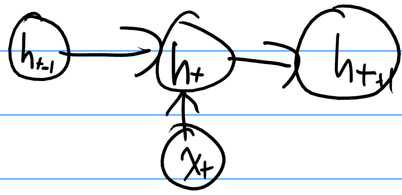
- Truncate gradient chains
  ↳ Stop grad comps after τ steps

- Gradien clipping
  If $\|g\|_2 > v$ ← threshold
  then $g \leftarrow \frac{g v}{\|g\|}$

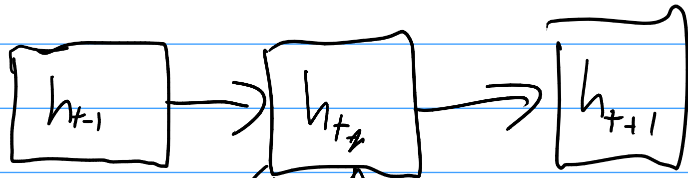


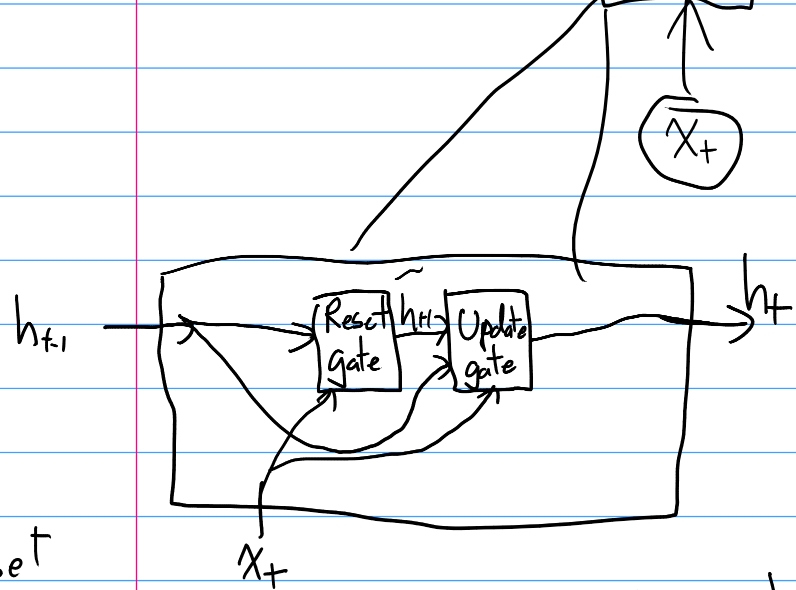
Gated Recurrent Units (GRUs)

Before:



After:





Simplification:
Reset, Update are bools $=0$ or 1

If "update" $=1$:
$h_t = h_{t-1}$ (use old state)
Else:
If "reset" $=1$:
$h_t = \tanh(Ux_t + Wh_{t-1} + b)$
(Standard RNN update)
Else:
$h_t = \tanh(Ux_t + b)$
(Drop old state)
MLP

Reset
↓

$$\left.\begin{aligned} R_t &= \text{sigmoid}\left(U^{(r)}x_t + W^{(r)}h_{t-1} + b^{(r)}\right) \\ Z_t &= \text{sigmoid}\left(U^{(z)}x_t + W^{(z)}h_{t-1} + b^{(z)}\right) \end{aligned}\right\} \begin{array}{l}\text{Vectors } \in [0,1] \\ \text{coordinatewise} \\ \text{same dim} \\ \text{as } h_t\end{array}$$

↑
Update

$$\tilde{h}_t = \tanh(Ux_t + W(R_t \odot h_{t-1}) + b)$$

$\tilde{h}_t$: candidate

$\odot$: coordinatewise product

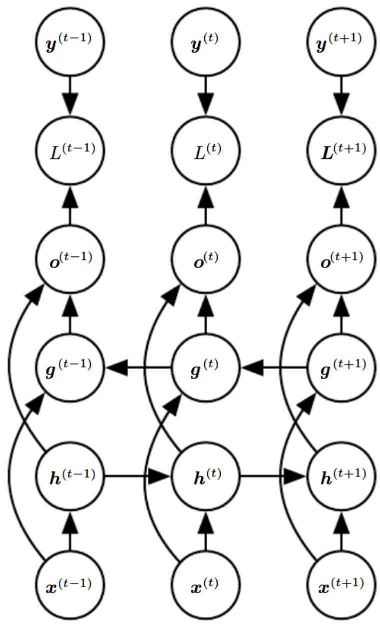
$$h_t = Z_t \odot h_{t-1} + (1 - Z_t) \odot \tilde{h}_t$$

Long Short Term Memory (LSTM)

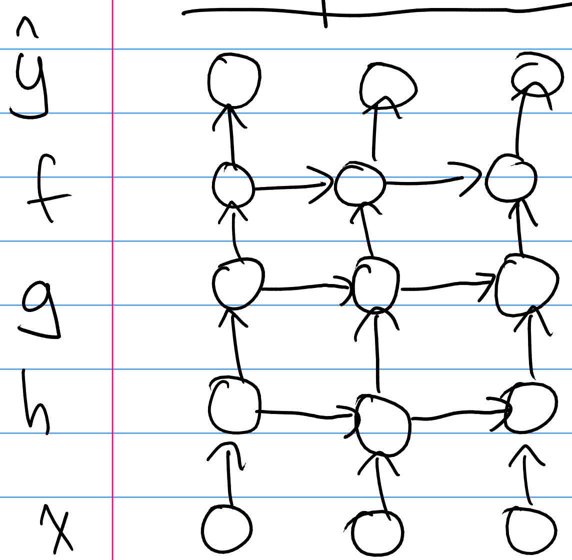
Bidirectional RNNs

named entity recognition

Eg.: I went to the bank
a) of the river
b) to withdraw cash

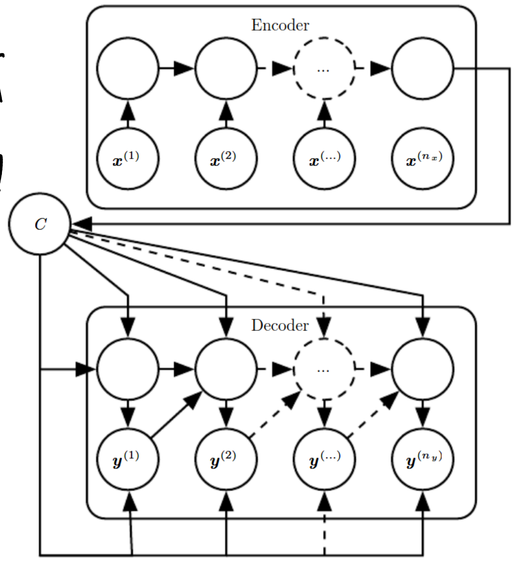


Deep RNNs



Encoder-Decoder

Context



Machine translation