RECURRENT NEURAL NETWORKS

Introduction to Deep Learning

CMPT 728/420

Oliver Schulte

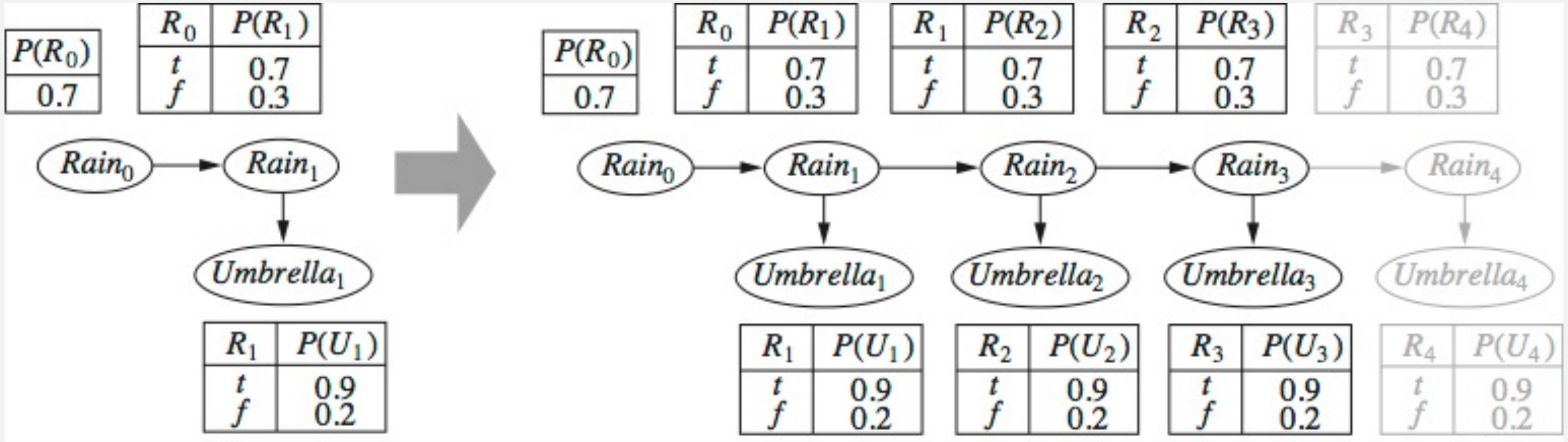
MODELS FOR SEQUENTIAL DATA

THE MARKOV ASSUMPTION

- Given the current (vector of) inputs, the next output is independent of the previous outputs. $P(y_t|\mathbf{x}_t,\mathbf{x}_{t-1},\ldots,\mathbf{x}_0)=P(y_t|\mathbf{x}_t)$.
- Example: Basketball Prediction (open in Chrome)
- *k*-order Markov process: next observation depends on fixed-length part of previous history.
  - sliding window model
  - convolutional neural net

MARKOV CHAIN MODEL

- At each point, the system is in a state s_t
- Given the state, the next output is independent of observations
  $P(y_t|s_t,\mathbf{x}_t,\mathbf{x}_{t-1},\ldots,\mathbf{x}_0)=P(y_t|s_t)$
- Current state depends only on current observation and previous state
  $P(s_t|s_{t-1},\mathbf{x}_t,\mathbf{x}_{t-1},\ldots,\mathbf{x}_0)=P(s_t|s_{t-1}, \mathbf{x}_t)$



HIDDEN MARKOV MODEL (HMM)

A Markov chain model where <u>the state is not observed</u>. Like a cluster.

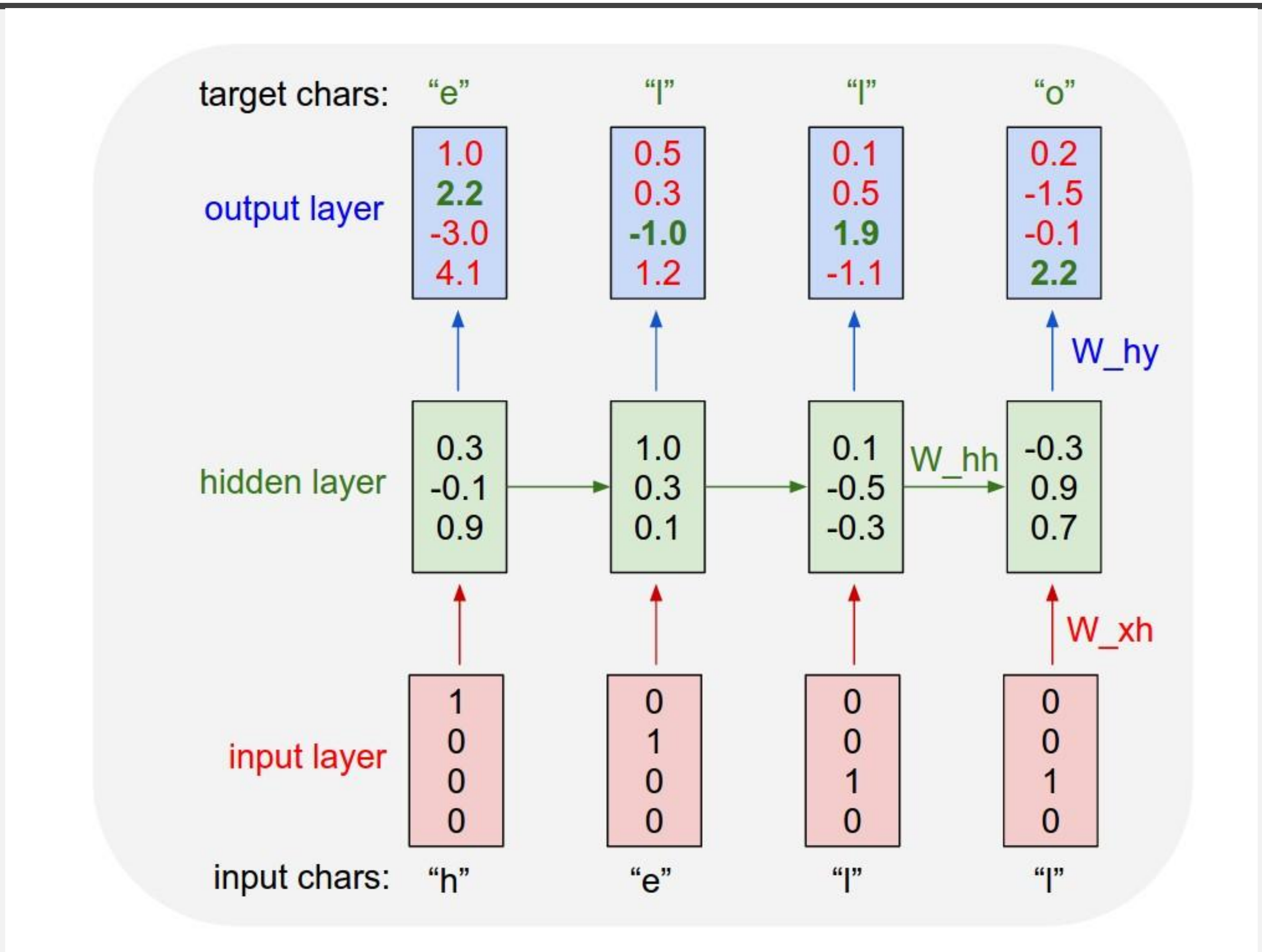
1. The output at time *t* is independent of previous inputs given the (right) hidden state at time *t*
   $P(y_t|h_t, \mathbf{x}_t, \mathbf{x}_{t-1}, \ldots, \mathbf{x}_0) = P(y_t|h_t)$.
2. The hidden state at time *t* is independent of previous inputs given the previous hidden state at time *t-1*
   $P(h_t|h_{t-1}, \mathbf{x}_t, \mathbf{x}_{t-1}, \ldots, \mathbf{x}_0) = P(h_t|h_{t-1})$

- Number of hidden states *k* is specified in advance.
- Hidden state ≈ cluster of history.

RECURRENT NEURAL NET (RNN)

- Basic Idea: Like HMM where
  Hidden State ➔ Activation Vector of hidden nodes.
- $P(y_t|\mathbf{h}_t, \mathbf{x}_t, \mathbf{x}_{t-1}, \ldots, \mathbf{x}_0) = P(y_t|\mathbf{h}_t)$ where $\mathbf{h}_t$ is the activation vector of hidden states
  $P(\mathbf{h}_t|\mathbf{h}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t-1}, \ldots, \mathbf{x}_0) = P(\mathbf{h}_t|\mathbf{h}_{t-1})$
- In terms of NN activations:
  $$h^i(t) = g_i\Big(\sum_m w_{im} h^m(t-1) + w_{ii} h^i(t-1)\Big)$$
  Hidden unit activation depends on its own previous activation
- If output = next observation, can use to generate sequences ($y_t = x_{t+1}$)
- Generation Demo

UNROLLED RNN



Rnn examples

LSTMS AND PROGRAMS

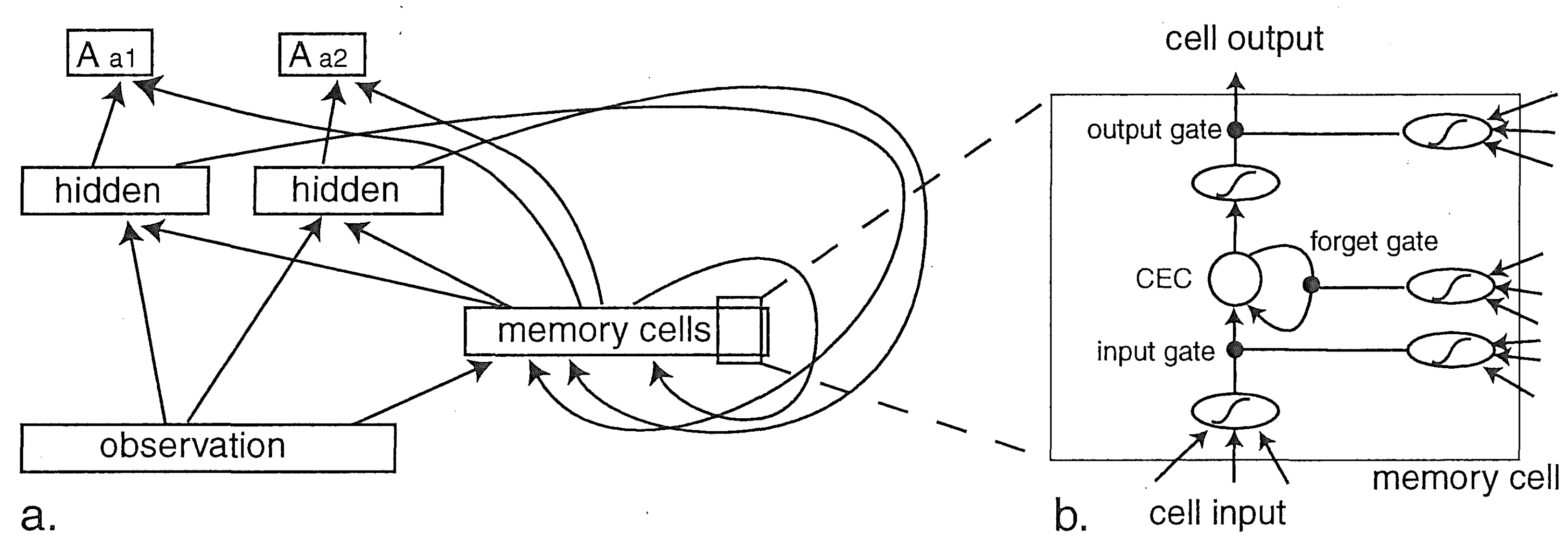
RNNS AND LONG-RANGE DEPENDENCIES

- Problem: The hidden units have to remember information. But when does past information become relevant to present prediction?
- Example: "In Korea, more than half of all the residents speak Korean".
- ➢ RNNs have trouble learning long-range dependencies
- More technically, we have the vanishing/exploding gradient problem in unrolling.
  - Roughly, temporal chain rule ➔ product of gradients
    - gradients < 1 ➔product close to 0
    - gradients > 1 ➔product explodes

LONG SHORT-TERM MEMORY (LSTM)

- Motivation: improve ability to learn long-range dependencies.
- Complicated Model, intuitions:
  - Introduce special hidden units called "memory cells".
  - Content of memory cells is carried from past to future on a "special track".
- More precisely: the current content of a memory cell has linear dependence on its previous content ➔ gradients neither vanish nor explode.
- What should be put into a memory cell? – "Input gates" learn to fill them.
- When is the content of a memory cell relevant? – "Output gates" learn when to use it.
- What if the content in our precious memory cells is no longer relevant? – "Forget gates" learn when to erase them.

LSTM CONNECTION DIAGRAM



MEMORY CELLS AND VARIABLES

- One way to think of a memory cell is that it is a probabilistic version of a variable in a traditional program.
- For a traditional program variable, you can assign it new values, retrieve the value when needed, update the value.
- A := 5
begin
....
end
if A > 4 then output ...

PROCESSING KOREAN EXAMPLE

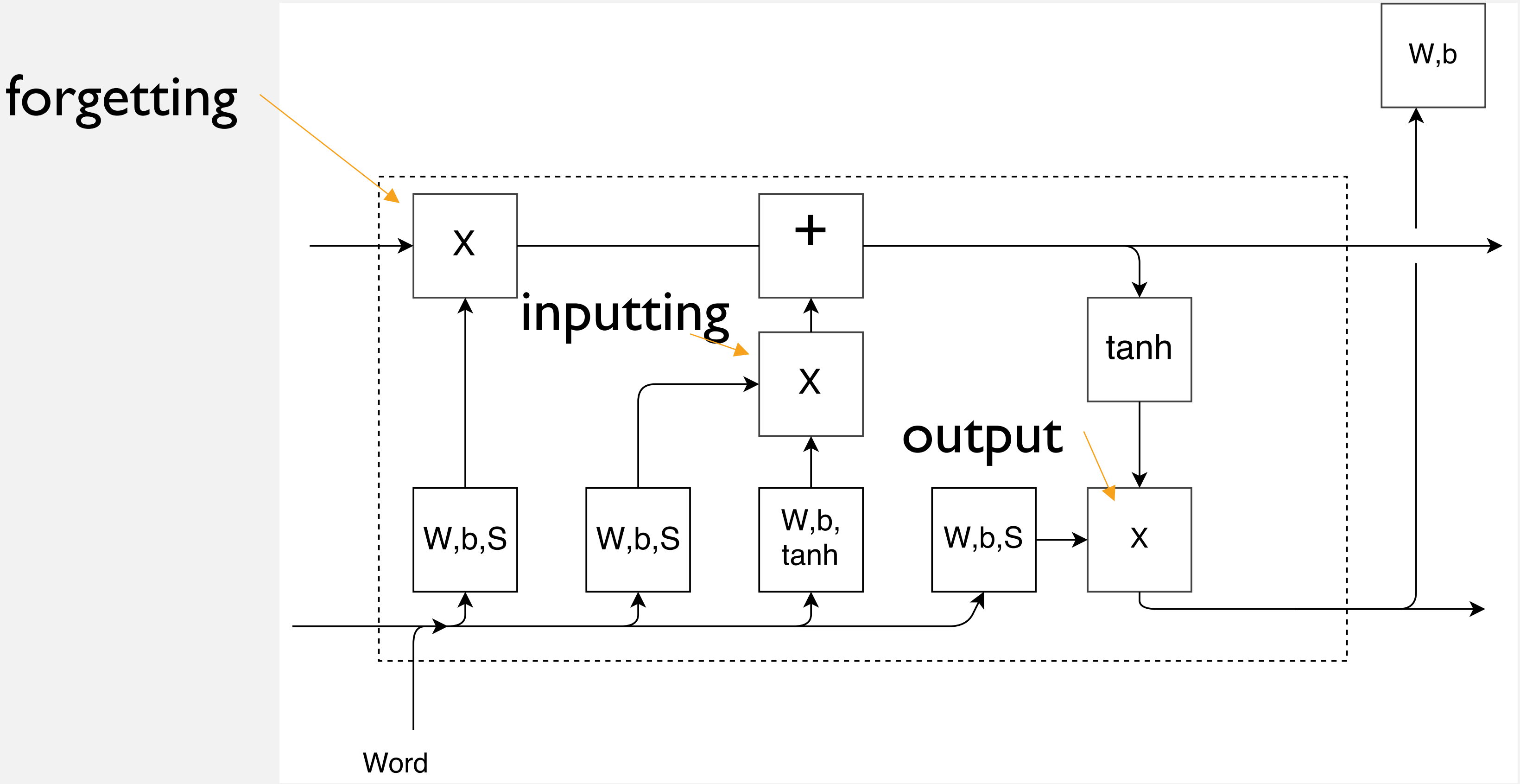
1. cell := empty /* initialize memory cell */
2. while not end_of_sequence
   1. current_word := read_next_word
   2. previous_cell := cell
   3. if input context is right
      cell := current_word /*e.g. after "In" store "Korea"
      else
      cell := previous_cell /*copy previous value*/
   4. if output context is right
      use cell to predict /* e.g. after "speak" predict "Korean" */

"In Korea, more than half of all the residents speak Korean".

LSTM PROGRAM WITH DETERMINISTIC GATES

1. cell := empty /* initialize memory cell */
2. while not end_of_sequence
    1. current_word := read_next_word
    2. previous_cell := cell
    3. compute input_gate, forget_gate, output_gate activations using previous hidden node activations
       /* like an RNN */
    4. compute candidate_memory using previous hidden node activations
       /*without using previous_cell */
       /* like an RNN */
    5. if input_gate is on
       cell := candidate_memory
       elseif forget_gate is on
       cell := empty
       else cell := previous_cell
    6. if output_gate is on, predict using current cell activation, current hidden node activations
       /* like an RNN */

INPUT AND FORGET GATES



SEMI-FORMAL DEFINITIONS (BAKKER 2001)

- Recall the RNN recurrence $h^i(t) = g_i(\sum_m w_{im} h^m(t-1) + w_{ii} h^i(t-1))$
- Applies to hidden units and input/output/forget gates (ignoring current input x_t for now).

- The new *memory cell contents* is

  [input factor x previous input activations] + [forgetting factor x previous content]

$$s^c(t) = [h^{input_c}(t) \times g_c(\sum_m w_{cm} h^m(t-1))] + [h^{\text{forget}_c}(t) \times s^c(t-1)]$$

- The new memory cell output is the output gate x cell activation

$$h^c(t) = h^{\text{out}_c}(t) \times g_c(s^c(t))$$

GATED RECURRENT UNIT

- Several variations on the LSTM gates have been developed.
- Especially popular is GRU
- Intuition: replace "input" and "forget" by "update"

GRU EQUATIONS

- New content $\mathbf{h'}_t = \tanh(W\mathbf{x}_t + \mathbf{r}_t \odot U\mathbf{h}_{t-1})$

Transform previous hidden state $\mathbf{h}_{t-1}$

Transform current observation $\mathbf{x}_t$

Reset gate for previous hidden state

- Update hidden state $\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \mathbf{h'}_t$

With probability $\mathbf{z}_t$ keep previous hidden state

With probability $1\text{-}\mathbf{z}_t$ adopt new content

- Reset gates $\mathbf{r}_t$ and update gates $\mathbf{z}_t$ are trainable function of previous hidden state and current input

CONCLUSION

- Hidden state at time t = summary of data up to time t
- Hidden layer in NN = distributed continuous representation of hidden state
- Recurrent NN: hidden state at time t-1 feeds into hidden state at time t
- Powerful method for learning with sequential data
- Problem: long-range dependencies and the vanishing gradient problem
- Possible solutions: gating with essentially linear updates (LSTMs, GRU)