Recurrent Network

11-785 Introduction to Deep Learning

- lecture 14 & 15 -

TAVE Research DL001

Heeji Won

Contents

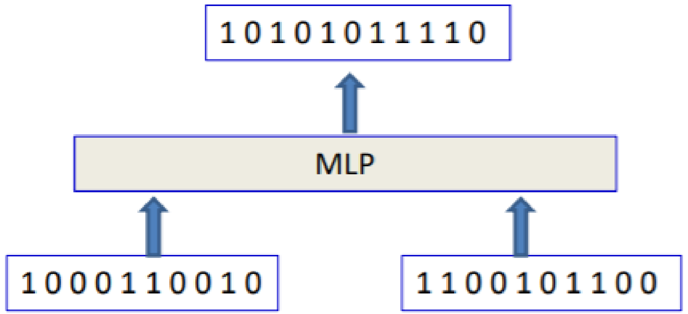
Contents

01. Benefits of RNN

- Examples

– Add two N-bit number to produce N+1-bit number

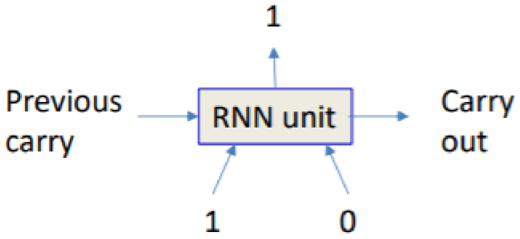
– Input is binary



✓ Require large number of training instances $(2^N \times 2^N)$

✓ not work for N+1 bit numbers

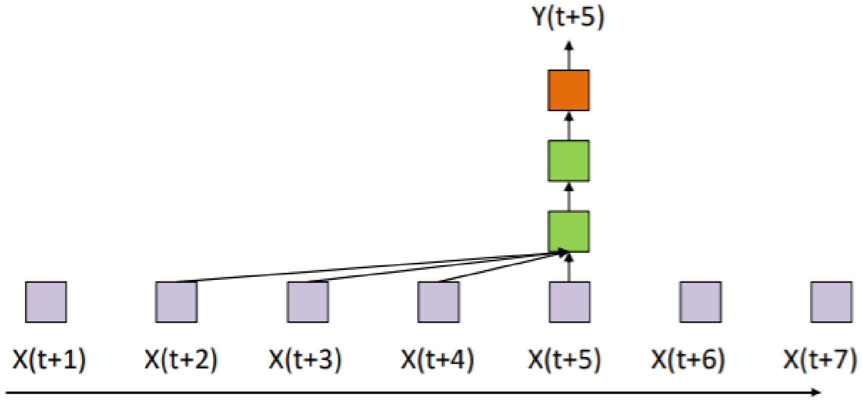
- How about RNN?



✓ Needs very little training data $(2 \times 2 \times 2)$

✓ Can add two numbers of any size

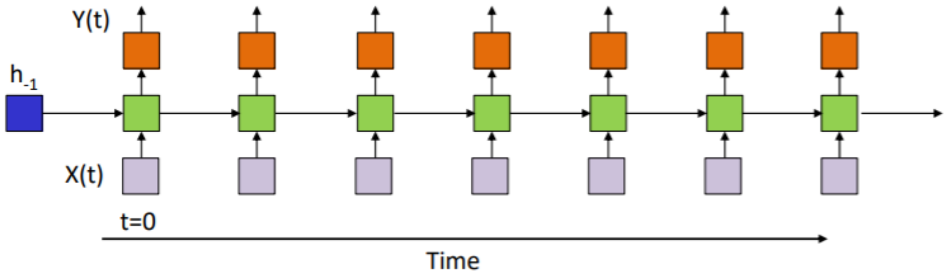
- RNN



Contents

02. Stability

- "BIBO" Stability
  - "Bounded Input Bounded Output" stability
  - Guaranteed if output and hidden activation are bounded
  - do



✓ But will it saturate

"Streetlight effect"

- Linear systems

$$h_k = W_h h_{k-1} + W_x x_k$$

$$= W_h^{k+1} h_{-1} + W_h^k W_x x_0 + W_h^{k-1} W_x x_1 + W_h^{k-2} W_x x_2 + \cdots$$

$$= H_k(h_{-1}) + H_k(x_0) + H_k(x_1) + H_k(x_2) + \cdots$$

$$= h_{-1} H_k(1_{-1}) + x_0 H_k(1_0) + x_1 H_k(1_1) + x_2 H_k(1_2) + \cdots$$

✓ Focus on second term $(W_h^k W_x x_0)$!

02. Stability

- The rate of blow up or vanishing

$$h_k = W_h^{k+1}h_{-1} + W_h^k W_x x_0 + W_h^{k-1}W_x x_1 + W_h^{k-2}W_x x_2 + \cdots$$

focus on this term

$$\boxed{W_h = U\Lambda U^{-1} \Leftrightarrow W_h u_i = \lambda_i u_i}$$

$$x' = W_x x = a_1u_1 + a_2u_2 + \cdots + a_nu_n$$

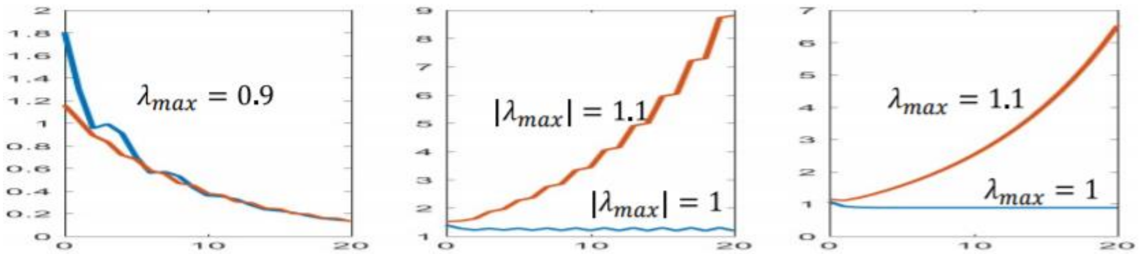
$$W_h x' = a_1\lambda_1u_1 + a_2\lambda_2u_2 + \cdots + a_n\lambda_nu_n$$

$$W_h^t x' = a_1\lambda_1^tu_1 + a_2\lambda_2^tu_2 + \cdots + a_n\lambda_n^tu_n$$

$$\lim_{t\to\infty}|W^t x'| = a_m\lambda_m^t u_m \quad where\ m = argmax_j\lambda_j$$

✓ If $|\lambda_{max}| > 1$ it will blow up, otherwise it will contract and shrink to 0 rapidly

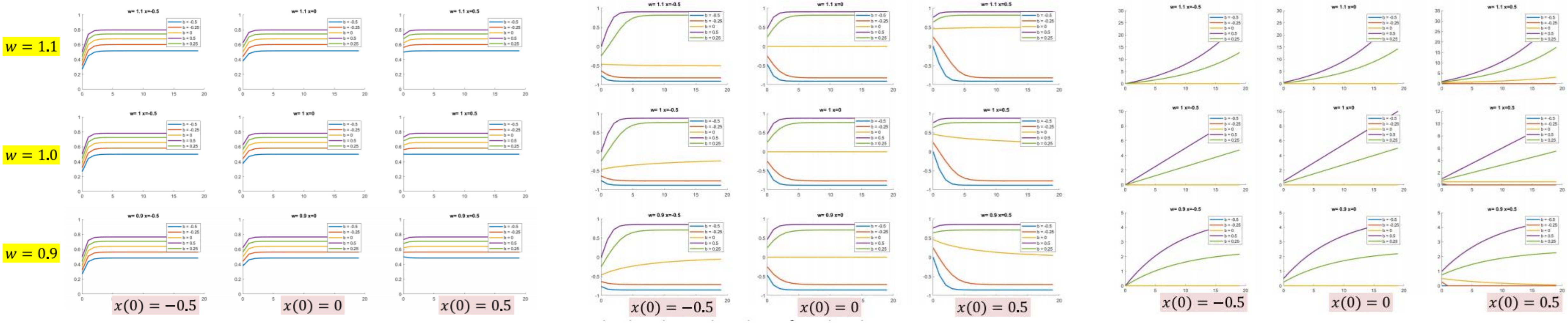
✓ The rate of blow up or vanishing depends only on the Eigen values (not on the input)



02. Stability

- How about non-linear activations?

<Sigmoid> <tanh> <RELU>

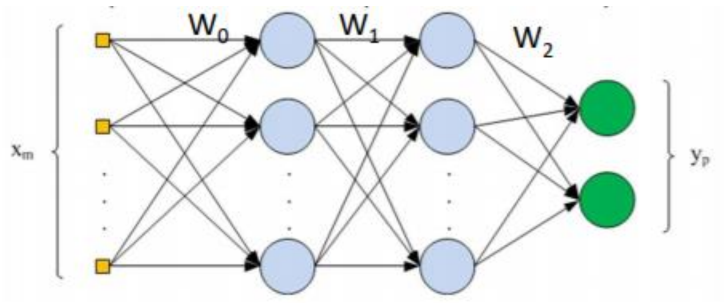


- Sigmoid activations saturate and the network becomes unable to retain new information
- RELU activations blow up or vanish rapidly
- Tanh activations are the slightly more effective at storing memory, but for not very long

Contents

03. Vanishing gradient problems

- Training deep networks

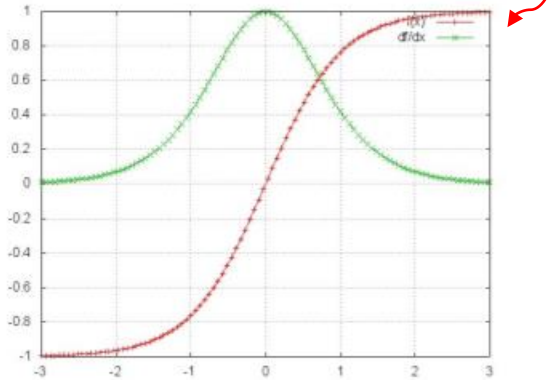


$$Y = f_N\left(W_N f_{N-1}\left(W_{N-1} f_{N-2}\left(\dots W_1 X\right)\right)\right)$$

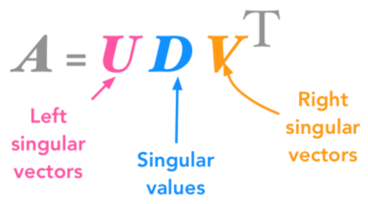
$$\nabla_{f_k} Div = \nabla D . \nabla f_N . W_N . \nabla f_{N-1} . W_{N-1} \dots \nabla f_{k+1} W_{k+1}$$

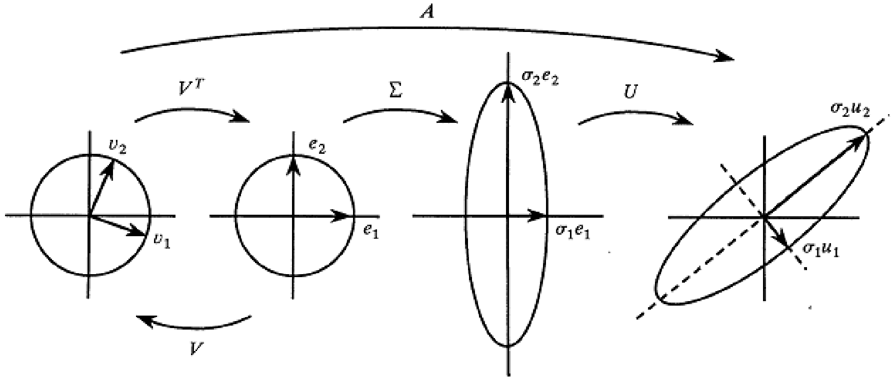
bounded

$$\nabla f_t(z_i) = \begin{bmatrix} f'_{t,1}(z_1) & 0 & \cdots & 0 \\ 0 & f'_{t,2}(z_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f'_{t,N}(z_N) \end{bmatrix}$$



$$\nabla_{f_k} Div = \nabla D . \nabla f_N . W_N . \nabla f_{N-1} . W_{N-1} \dots \nabla f_{k+1} W_{k+1}$$



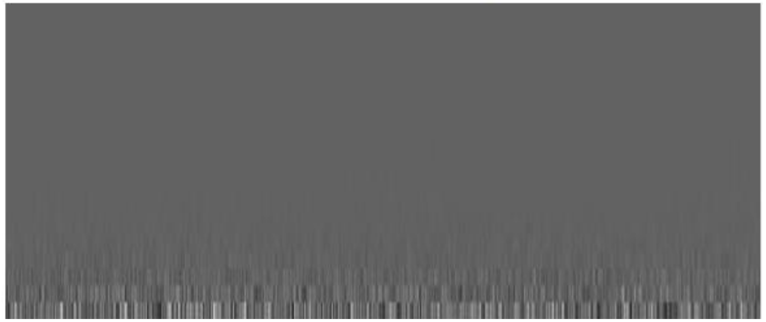


- Expand ∇D along directions in which the singular values of the weight matrices are greater than 1
- Shrink ∇D in directions where the singular values are less than 1

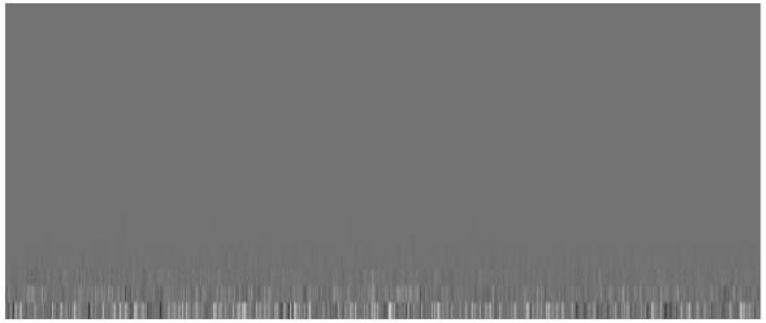
03. Vanishing gradient problems

- Vanishing gradient examples

ELU activation, Batch gradients



RELU activation, Batch gradients

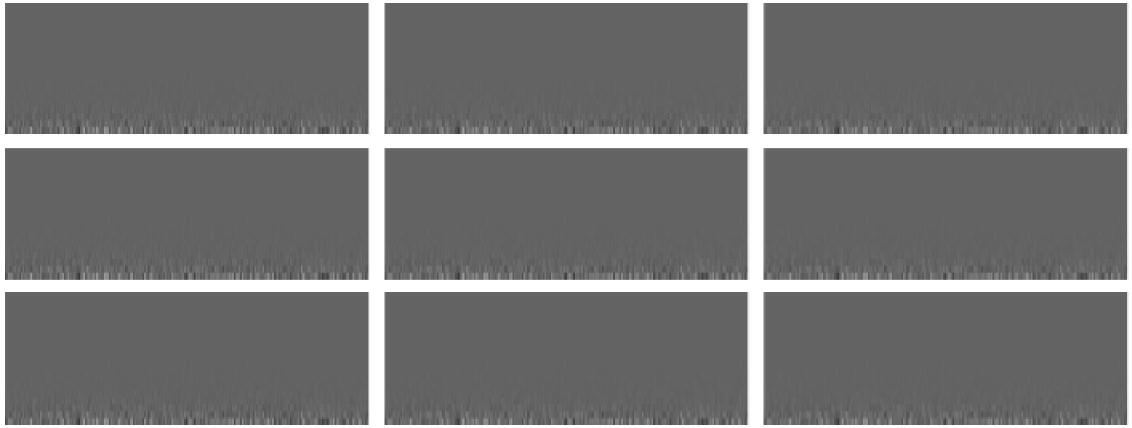


Sigmoid activation, Batch gradients

Tanh activation, Batch gradients

- ELU activations maintain gradients longest
- But in all cases gradient effectively vanish after 10 layers

ELU activation, Individual instances



Problems of RNN

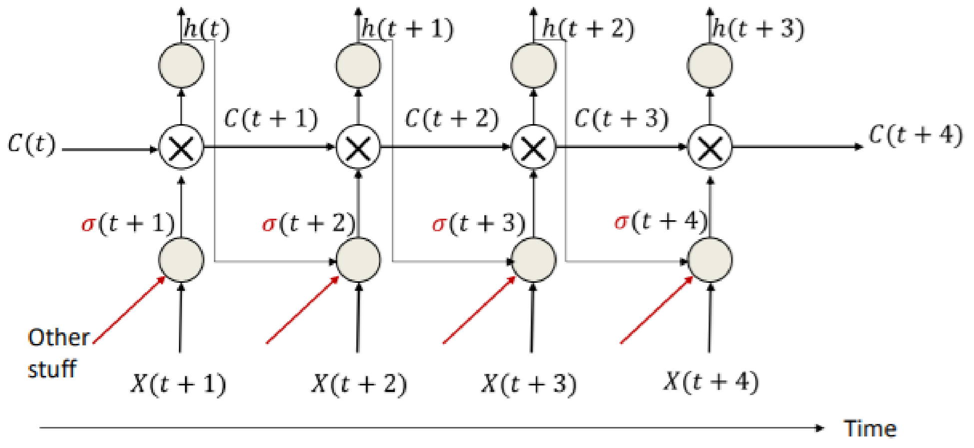
✓ Vanishing gradient problems
✓ Loss of memories over time

Contents

04. LSTMs and variants

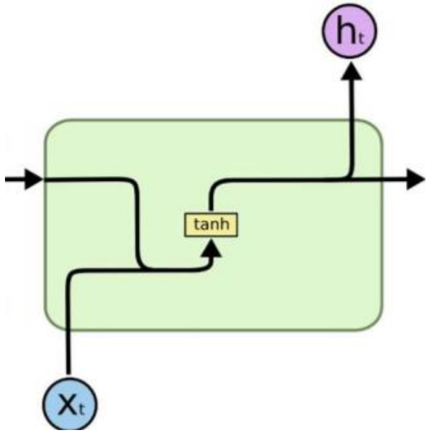
- LSTM' s basic idea

– the constant error carousel

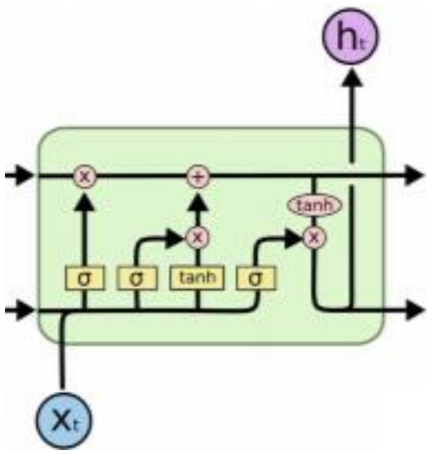


– The gate σ depends on current input, current hidden state and so one

- LSTM

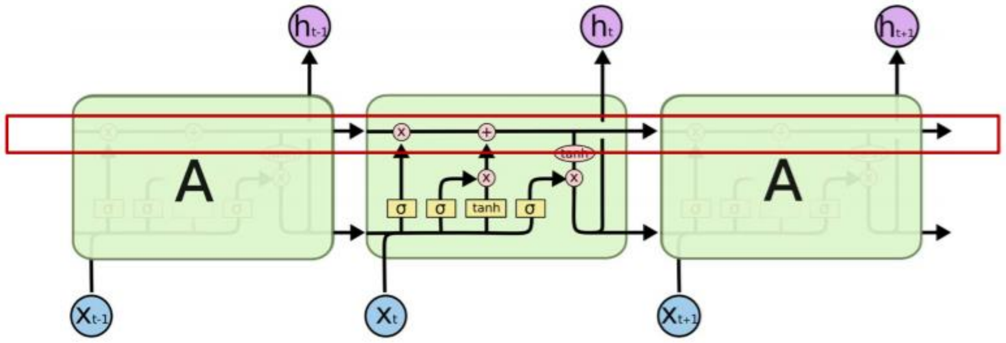


<Standard RNN>



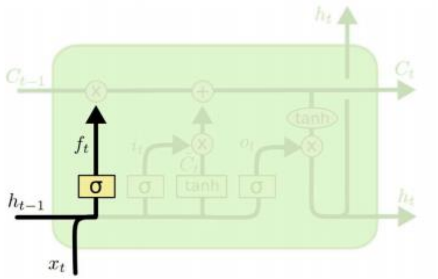
<LSTM>

- Constant Error Carousel



04. LSTMs and variants

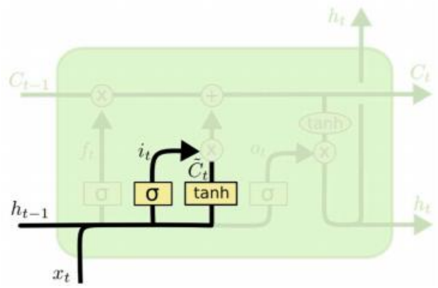
- Forget gate



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

- determines whether to carry over the history or to forget it

- Input gate

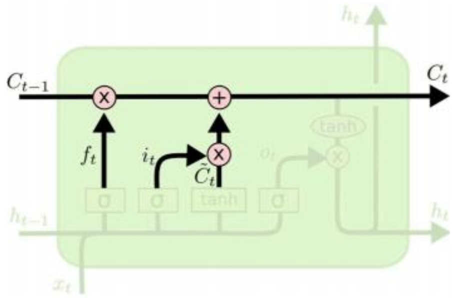


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

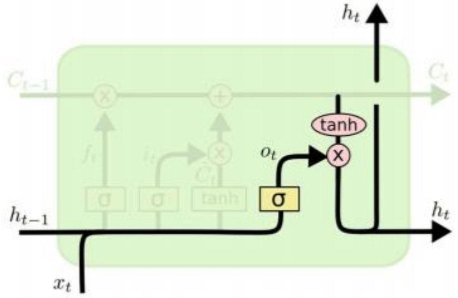
- A layer determines if there' s something new in the input
- A gate decides if its worth remembering

- Memory cell update



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- Output gate

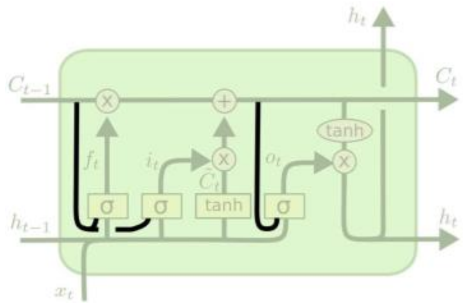


$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

compressed

➢ Peephole connection



using both C and h

$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_i\right)$$

$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \ + \ b_o\right)$$

04. LSTMs and variants

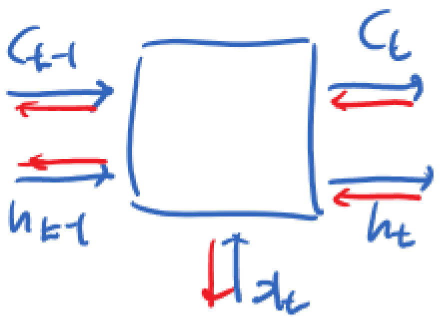
- Forward

Gates

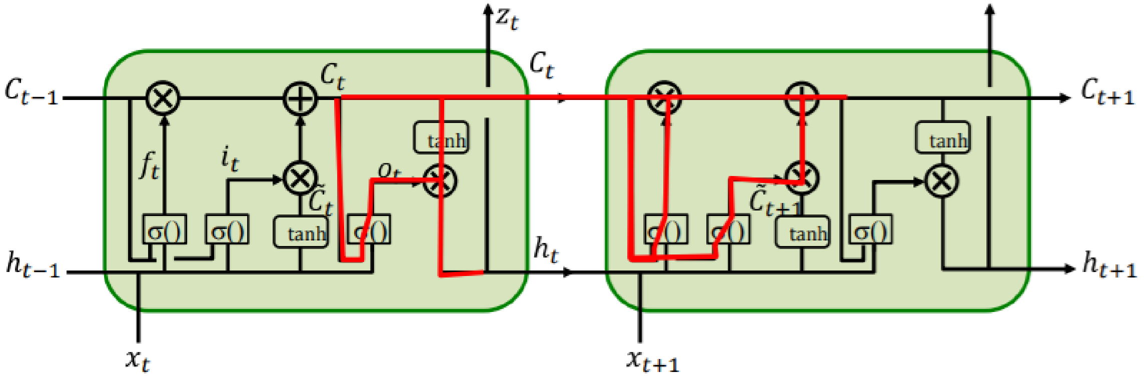
$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \ + \ b_o\right)$$

Variables

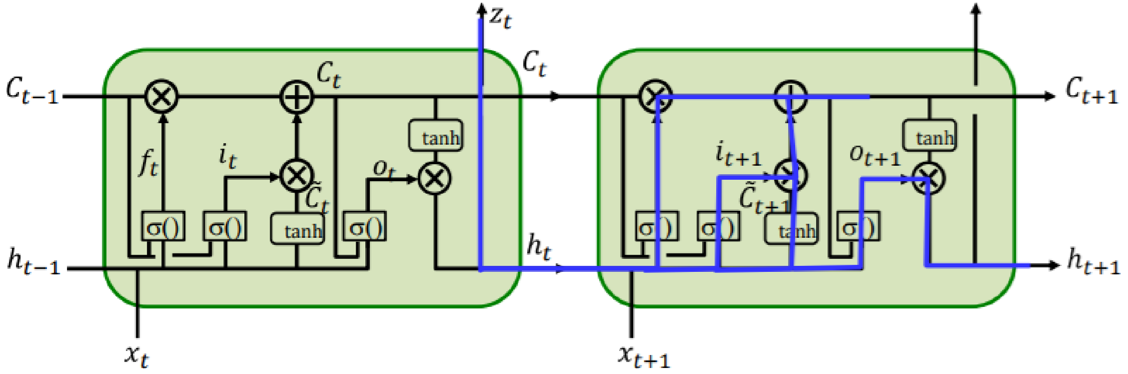
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$h_t = o_t * \tanh(C_t)$$



- backward



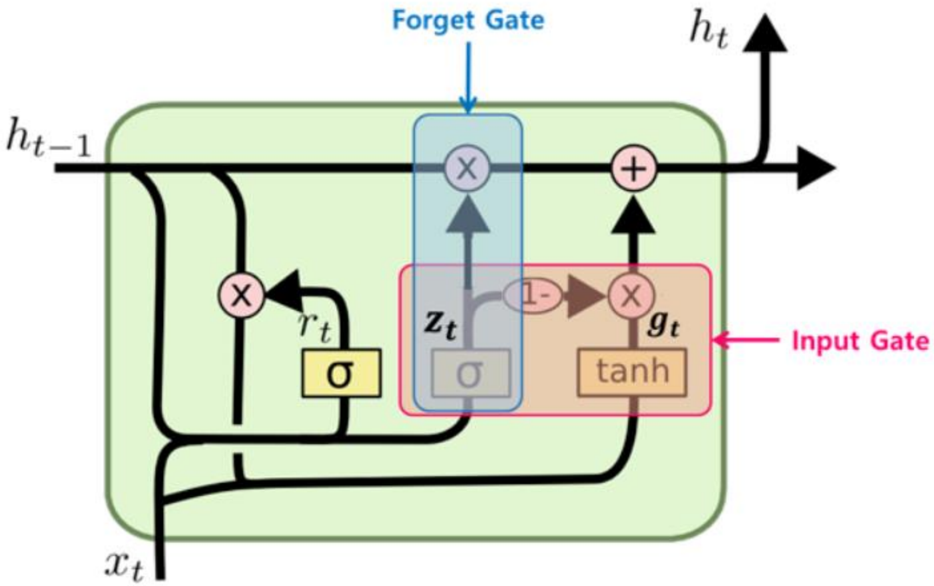
$$\nabla_{C_t} Div = \nabla_{h_t} Div \circ (o_t \circ tanh'(.) + tanh(.) \circ \sigma'(.) W_{Co}) +$$
$$\nabla_{C_{t+1}} Div \circ \left(f_{t+1} + C_t \circ \sigma'(.) W_{Cf} + \tilde{C}_{t+1} \circ \sigma'(.) W_{Ci} \circ \tanh(.) \ldots\right)$$



$$\nabla_{h_t} Div = \nabla_{z_t} Div \nabla_{h_t} z_t + \nabla_{C_{t+1}} Div \circ \left(C_t \circ \sigma'(.) W_{hf} + \tilde{C}_{t+1} \circ \sigma'(.) W_{hi}\right) +$$
$$\nabla_{C_{t+1}} Div \circ o_{t+1} \circ tanh'(.) W_{hi} + \nabla_{h_{t+1}} Div \circ tanh(.) \circ \sigma'(.) W_{ho}$$

04. LSTMs and variants

- GRU
  - Simplified version of LSTM



$$\mathbf{r}_t = \sigma\left(\mathbf{W}_{xr}^T \cdot \mathbf{x}_t + \mathbf{W}_{hr}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_r\right)$$
$$\mathbf{z}_t = \sigma\left(\mathbf{W}_{xz}^T \cdot \mathbf{x}_t + \mathbf{W}_{hz}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_z\right)$$
$$\mathbf{g}_t = \tanh\left(\mathbf{W}_{xg}^T \cdot \mathbf{x}_t + \mathbf{W}_{hg}^T \cdot (\mathbf{r}_t \otimes \mathbf{h}_{t-1}) + \mathbf{b}_g\right)$$
$$\mathbf{h}_t = \mathbf{z}_t \otimes \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \otimes \mathbf{g}_t$$

✓ One cell state, h_t

✓ Two gates, Reset gate(r_t) and Update gate(z_t)

✓ Reset gate control how much information of h_{t-1} will be considered

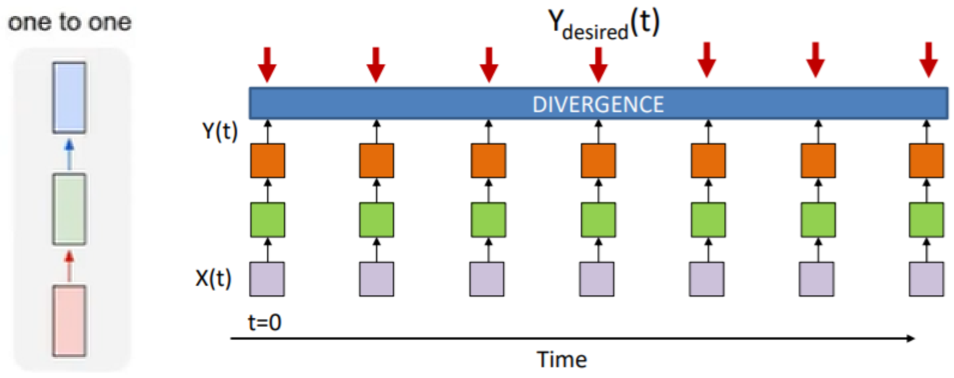
✓ Update gate control forget gate and input gate

- If output of z_t is 1, open **forget** gate and close input gate
- If output of z_t is 0, close forget gate and open **input** gate

Contents

05. Loss functions for RNN

- Regular MLP



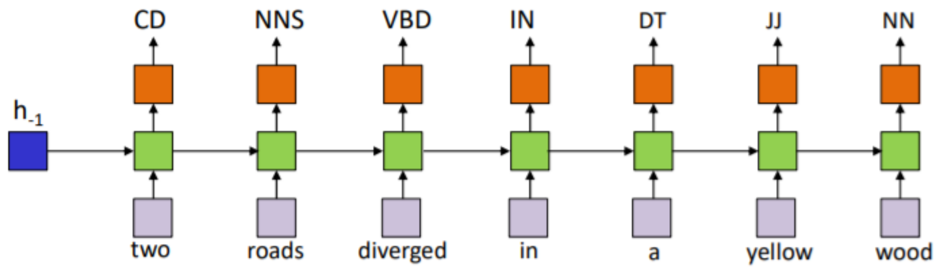
  - No recurrence
  - The output at time t is related to the output at $t' \neq t$

$$Div(Y_{target}(1 \dots T), Y(1 \dots T)) = \sum_t w_t Div(Y_{target}(t), Y(t))$$

$$\nabla_{Y(t)} Div(Y_{target}(1 \dots T), Y(1 \dots T)) = w_t \nabla_{Y(t)} Div(Y_{target}(t), Y(t))$$

Typical Divergence for classification: $Div(Y_{target}(t), Y(t)) = KL(Y_{target}(t), Y(t))$

- Time synchronous network



  - Input : symbols as one-hot vectors
  - Output : Probability distribution over symbols

$$Y(t,i) = P(V_i | w_0 \dots w_{t-1})$$

  - Training : BPTT
  - Divergence

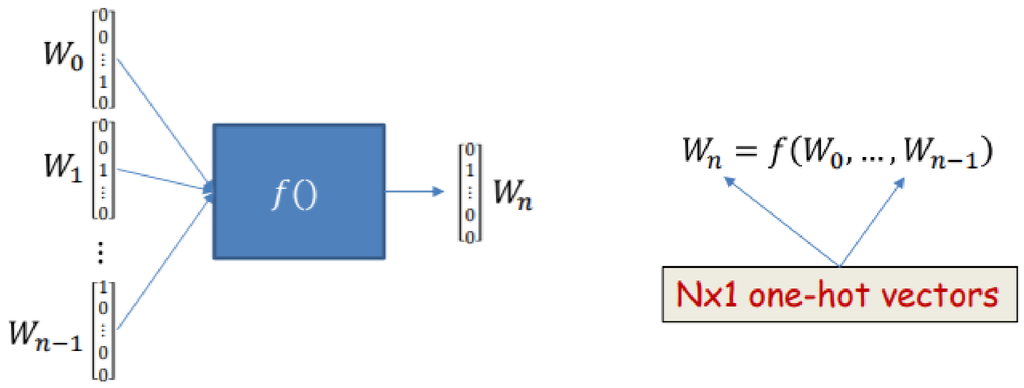
$$Div(Y_{target}(1 \dots T), Y(1 \dots T)) = \sum_t Div(Y_{target}(t), Y(t)) = -\sum_t \log Y(t, w_{t+1})$$

$$\nabla_{Y(t)} Div(Y_{target}(1 \dots T), Y(1 \dots T)) = \nabla_{Y(t)} Div(Y_{target}(t), Y(t))$$

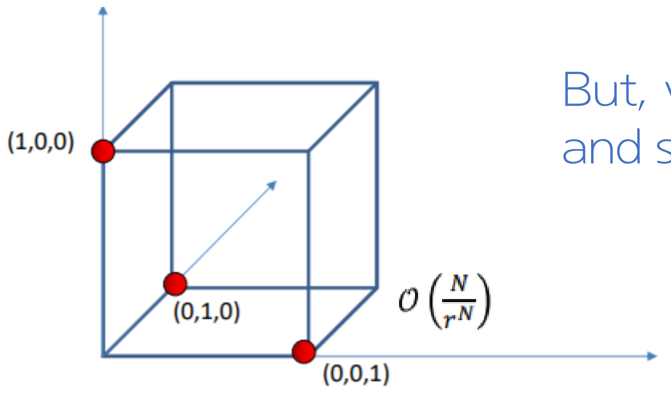
Contents

06. Sequence prediction

- One-hot vectors



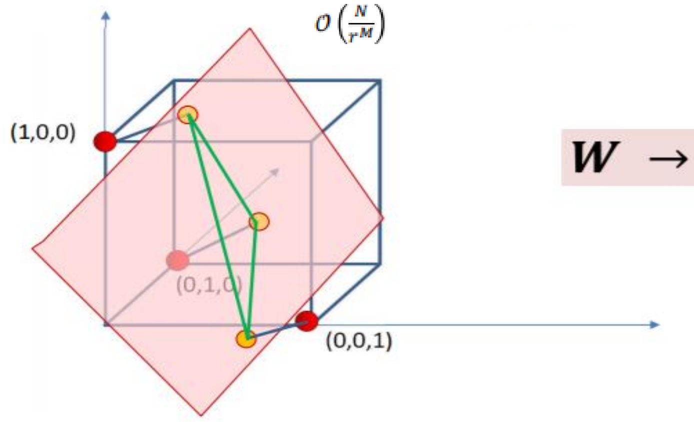
- makes no assumptions about relative importance and relationships



But, very high-dimension and sparse

wasteful!

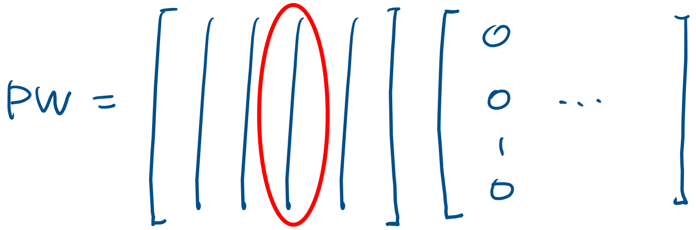
- Projection



$$\boldsymbol{W} \rightarrow \boldsymbol{PW}$$

learn a subspace plane which capture sematic relations between the words

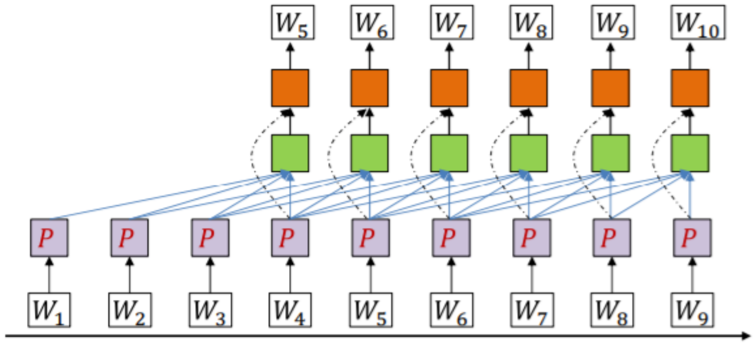
embedding vector



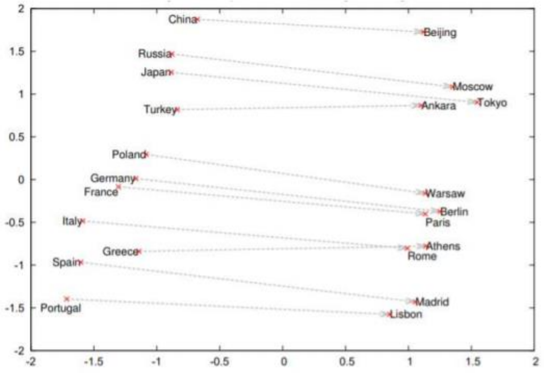
linear transform

06. Sequence prediction

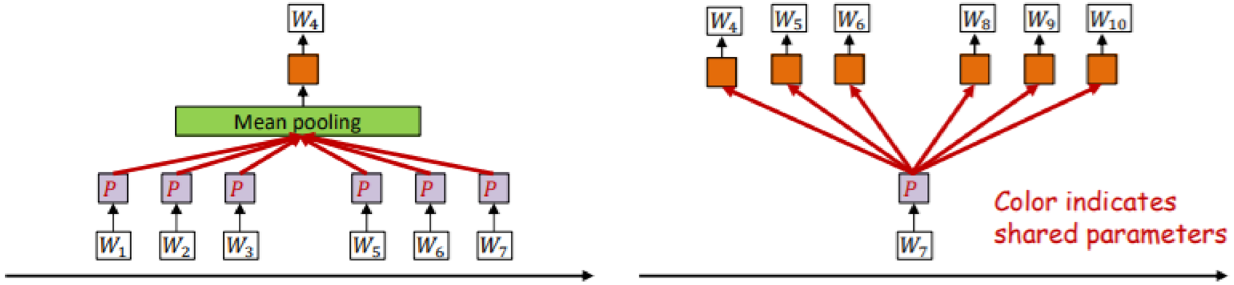
- The TDNN model



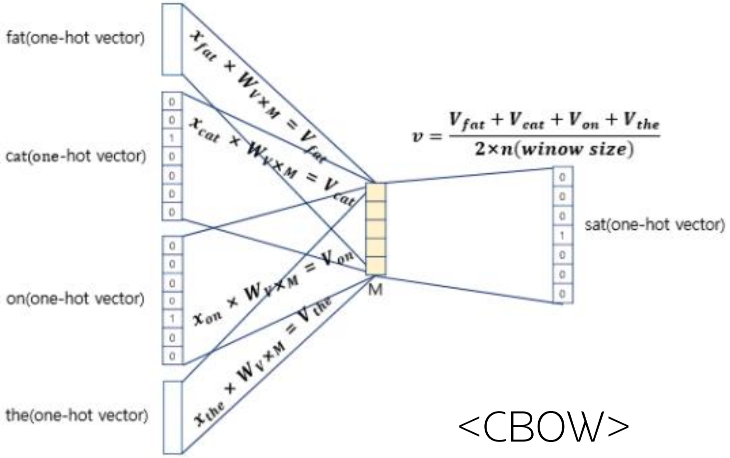
- predict each word based on the past N words
- also learn low-dimensional representations PW of words
- examples



- Alternative models to learn projections



- Soft bag of words : Predict word based on words in immediate context
- Skip-grams: Predict adjacent words based on current word



<CBOW>

Thank you