Recurrent Neural Networks 2

CS 287

(Based on Yoav Goldberg's notes)

Review: Representation of Sequence

- Many tasks in NLP involve sequences

$$w_1, \dots, w_n$$

- Representations as matrix dense vectors $\mathbf{X}$
  (Following YG, slight abuse of notation)

$$\mathbf{x}_1 = \mathbf{x}_1^0 \mathbf{W}^0, \dots, \mathbf{x}_n = \mathbf{x}_n^0 \mathbf{W}^0$$

- Would like fixed-dimensional representation.

Review: Sequence Recurrence

- Can map from dense sequence to dense representation.
- $\mathbf{x}_1, \ldots, \mathbf{x}_n \mapsto \mathbf{s}_1, \ldots, \mathbf{s}_n$
- For all $i \in \{1, \ldots, n\}$

$$\mathbf{s}_i = R(\mathbf{s}_{i-1}, \mathbf{x}_i; \theta)$$

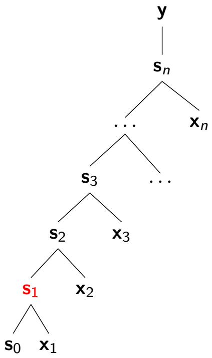
- θ is shared by all R

Example:

$$\begin{aligned}
\mathbf{s}_4 &= R(\mathbf{s}_3, \mathbf{x}_4) \\
&= R(R(\mathbf{s}_2, \mathbf{x}_3), \mathbf{x}_4) \\
&= R(R(R(R(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3), \mathbf{x}_4)
\end{aligned}$$

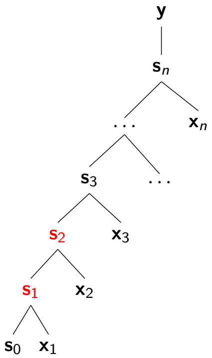
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



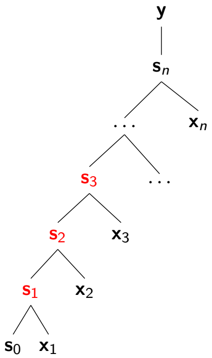
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



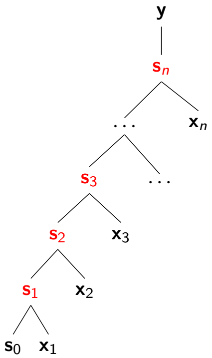
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



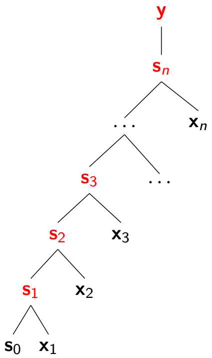
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



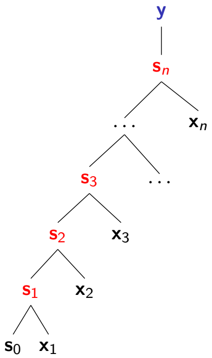
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



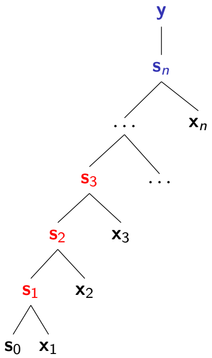
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



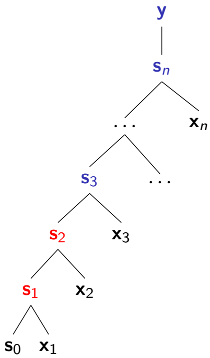
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



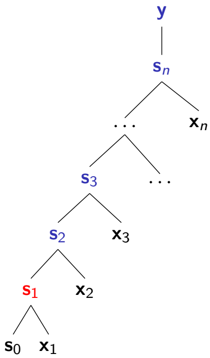
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



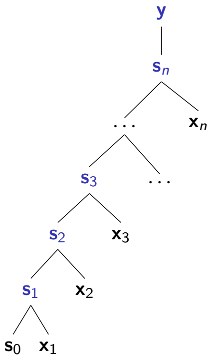
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



Issues

- Can be inefficient, but batch/GPUs help.
- Model is much deeper than previous approaches.
  - This matters a lot, focus of next class.
- Variable-size model for each sentence.
  - Have to be a bit more clever in Torch.

Quiz

Consider a ReLU version of the Elman RNN with function R defined as

$$NN(\mathbf{x}, \mathbf{s}) = \text{ReLU}(\mathbf{s}\mathbf{W}^s + \mathbf{x}\mathbf{W}^x + \mathbf{b}).$$

We use this RNN with an acceptor architecture over the sequence $\mathbf{x}_1, \ldots, \mathbf{x}_5$. Assume we have computed the gradient for the final layer

$$\frac{\partial L}{\partial \mathbf{s}_5}$$

What is the symbolic gradient of the previous state $\frac{\partial L}{\partial \mathbf{s}_4}$?

What is the symbolic gradient of the first state $\frac{\partial L}{\partial \mathbf{s}_1}$?

Answer

Chain rule, then relu cases, then to indicator notation

$$\begin{aligned}
\frac{\partial L}{\partial s_{4,i}} &= \sum_j \frac{\partial s_{5,j}}{\partial s_{4,i}} \frac{\partial L}{\partial s_{5,j}} \\
&= \sum_j \begin{cases} W^s_{i,j} \frac{\partial L}{\partial s_{5,j}} & s_{5,j} > 0 \\ 0 & o.w. \end{cases} \\
&= \sum_j \mathbf{1}(s_{5,j} > 0) W^s_{i,j} \frac{\partial L}{\partial s_{5,j}}
\end{aligned}$$

Answer

Multiple applications of Chain rule, combine relu cases.

$$
\begin{aligned}
\frac{\partial L}{\partial s_{1,j_1}} &= \sum_{j_2} \dots \sum_{j_5} \frac{\partial s_{5,j_5}}{\partial s_{4,j_4}} \frac{\partial L}{\partial s_{5,j_5}} \\
&= \sum_{j_2} \dots \sum_{j_5} \mathbf{1}(s_{2,j_2} > 0 \wedge \dots \wedge s_{5,j_5} > 0) W^s_{j_1,j_2} \dots W^s_{j_4,j_5} \frac{\partial L}{\partial s_{5,j_5}} \\
&= \sum_{j_2 \dots j_5} \prod_{k=2}^{5} \mathbf{1}(s_{k,j_k} > 0) W^s_{j_{k-1},j_k} \frac{\partial L}{\partial s_{5,j_5}}
\end{aligned}
$$

The Promise of RNNs

- We hope to learn a model with memory.
- For acceptors this means long-range interaction.

  `How can you not see this movie?`

  `You should not see this movie.`
- Memory interaction here is at $\mathbf{s}_1$, but gradient signal is at $\mathbf{s}_n$

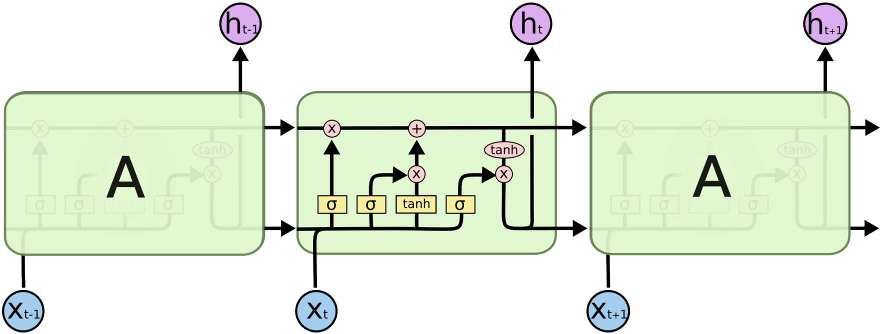
Vanishing Gradients

- Gradients at early layers go through many squashing layers.
- For instance consider quiz with hardtanh

$$\sum_{j_2 \ldots j_5} \prod_{k=2}^{5} \mathbf{1}(1 > s_{k,j_k} > 0) W^s_{j_{k-1},j_k} \frac{\partial L}{\partial s_{5,j_5}}$$

- The indicator term causes a tendency towards *vanishing gradients*.
- If this occurs, model cannot learn long-term dependencies.

LSTM (Hochreiter and Schmidhuber, 1997)



$$
\begin{aligned}
R(\mathbf{s}_{i-1}, \mathbf{x}_i) &= [\mathbf{c}_i, \mathbf{h}_i] \\
\mathbf{c}_i &= \mathbf{j} \odot \mathbf{i} + \mathbf{f} \odot \mathbf{c}_{i-1} \\
\mathbf{h}_i &= \tanh(\mathbf{c}_i) \odot \mathbf{o} \\
\mathbf{i} &= \tanh(\mathbf{x}\mathbf{W}^{xi} + \mathbf{h}_{i-1}\mathbf{W}^{hi} + \mathbf{b}^{i}) \\
\mathbf{j} &= \sigma(\mathbf{x}\mathbf{W}^{xj} + \mathbf{h}_{i-1}\mathbf{W}^{hj} + \mathbf{b}^{j}) \\
\mathbf{f} &= \sigma(\mathbf{x}\mathbf{W}^{xf} + \mathbf{h}_{i-1}\mathbf{W}^{hf} + \mathbf{b}^{f}) \\
\mathbf{o} &= \tanh(\mathbf{x}\mathbf{W}^{xo} + \mathbf{h}_{i-1}\mathbf{W}^{ho} + \mathbf{b}^{o})
\end{aligned}
$$

- $\mathbf{f}$; forget gate
- $\mathbf{i}$; input gate
- $\mathbf{c}$; cell state
- $\mathbf{h}$; hidden state

Contents

Deep Networks

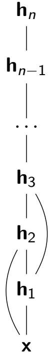
- This same issue occurs in deep MLPs.

$$NN_{layer}(\mathbf{x}) = \text{ReLU}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)$$

$$\begin{array}{c} \mathbf{h}_n \\ | \\ \mathbf{h}_{n-1} \\ | \\ \cdots \\ | \\ \mathbf{h}_2 \\ | \\ \mathbf{h}_1 \\ | \\ \mathbf{x} \end{array}$$

Thought Experiment: Additive Skip-Connections

$$NN_{sl1}(\mathbf{x}) = \frac{1}{2}\,\mathrm{ReLU}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1) + \frac{1}{2}\mathbf{x}$$



Exercise

Original model has same gradient issue as with RNN.

$$\frac{\partial L}{\partial h_{n-1,j_{n-1}}} = \sum_{j_n} \mathbf{1}(h_{n,j_n} > 0) W_{j_{n-1},j_n} \frac{\partial L}{\partial h_{n,j_n}}$$

Exercise: What happens to the gradient of $n-1$ with skip-connections ?

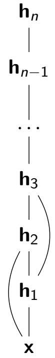
Exercise

We now have the average of two terms. One with no saturation condition.

$$\frac{\partial L}{\partial h_{n-1,j_{n-1}}} = \frac{1}{2}\Big(\sum_{j_n} \mathbf{1}(h_{n,j_n} > 0) W_{j_{n-1},j_n} \frac{\partial L}{\partial h_{n,j_n}}\Big) + \frac{1}{2}\Big(h_{n-1,j_{n-1}} \frac{\partial L}{\partial h_{n,j_{n-1}}}\Big)$$

Thought Experiment: Dynamic Skip-Connections

$$
\begin{aligned}
NN_{sl2}(\mathbf{x}) &= (1-t)\,\mathrm{ReLU}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1) + t\mathbf{x} \\
t &= \sigma(\mathbf{x}\mathbf{W}^t + b^t) \\
\mathbf{W}^t &\in \mathbb{R}^{d_{\mathrm{in}} \times 1}
\end{aligned}
$$



Thought Experiment: Dynamic Skip-Connections

$$
\begin{aligned}
NN_{sl2}(\mathbf{x}) &= (1-t)\,\mathrm{ReLU}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1) + t\mathbf{x} \\
t &= \sigma(\mathbf{x}\mathbf{W}^t + b^t) \\
\mathbf{W}^t &\in \mathbb{R}^{d_{\mathrm{hid}} \times 1}
\end{aligned}
$$

The t values are saved on the forward pass.

$$
\begin{aligned}
\frac{\partial L}{\partial h_{n-1,j_{n-1}}} = \quad & (1-t) \quad (\sum_{j_n} \mathbf{1}(h_{n,j_n} > 0) W_{j_{n-1},j_n} \frac{\partial L}{\partial h_{n,j_n}}) \\
+ \quad & \quad t \qquad (h_{n-1,j_{n-1}} \frac{\partial L}{\partial h_{n,j_{n-1}}})
\end{aligned}
$$

Thought Experiment: Dynamic Skip-Connections

- Note: $\mathbf{W}^t$ is also receiving gradients through the sigmoid!
- Learn how to trade-off skipping versus deep layers.
- (Backprop is fun.)

$$
\begin{aligned}
NN_{sl2}(\mathbf{x}) &= (1-t)\,\mathrm{ReLU}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1) + t\mathbf{x} \\
t &= \sigma(\mathbf{x}\mathbf{W}^t + b^t) \\
\mathbf{W}^t &\in \mathbb{R}^{d_{\mathrm{hid}} \times 1}
\end{aligned}
$$

Highway Network (Srivastava et al., 2015)

$$
\begin{aligned}
NN_{highway}(\mathbf{x}) &= (1 - \mathbf{t}) \odot \tilde{\mathbf{h}} + \mathbf{t} \odot \mathbf{x} \\
\tilde{\mathbf{h}} &= \mathrm{ReLU}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1) \\
\mathbf{t} &= \sigma(\mathbf{x}\mathbf{W}^t + \mathbf{b}^t) \\
\mathbf{W}^t &\in \mathbb{R}^{d_{\mathrm{hid}} \times d_{\mathrm{hid}}} \\
\mathbf{b}^t &\in \mathbb{R}^{1 \times d_{\mathrm{hid}}}
\end{aligned}
$$

- $\tilde{\mathbf{h}}$; *transform* (e.g. standard MLP layer)
- $\mathbf{t}$; *carry* (dimension-specific dynamic skipping)

Highway Gradients

The **t** values are saved on the forward pass.

$$\begin{aligned}\frac{\partial L}{\partial h_{n-1,j_{n-1}}} = & \qquad (\sum_{j_n}(1-t_{j_n})\mathbf{1}(h_{n,j_n}>0)W_{j_{n-1},j_n}\frac{\partial L}{\partial h_{n,j_n}}) \\ + \quad t_{j_{n-1}} & \quad (h_{n-1,j_{n-1}}\frac{\partial L}{\partial h_{n,j_{n-1}}})\end{aligned}$$

Gating

- This is known as the *gating* operation

$$\mathbf{t} \odot \mathbf{x}$$

- Allows vector $\mathbf{t}$ to mask or gate $\mathbf{x}$.
- True gating would have $\mathbf{t} \in \{0, 1\}^{d_{\mathrm{hid}}}$
- Approximate with the sigmoid,

$$\mathbf{t} = \sigma(\mathbf{W}^{t}\mathbf{x}\mathbf{b})$$

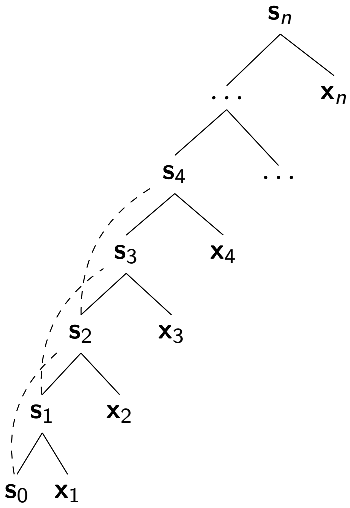
Contents

Back To RNNs

- Acceptor RNNs are deep networks with shared weights.
- Can replace Elman layer with modified highway layer.

$$
\begin{aligned}
R(\mathbf{s}_{i-1}, \mathbf{x}_i) &= (1 - \mathbf{t}) \odot \tilde{\mathbf{h}} + \mathbf{t} \odot \mathbf{s}_{i-1} \\
\tilde{\mathbf{h}} &= \tanh(\mathbf{x}\mathbf{W}^x + \mathbf{s}_{i-1}\mathbf{W}^s + \mathbf{b}) \\
\mathbf{t} &= \sigma(\mathbf{x}\mathbf{W}^{xt} + \mathbf{s}_{i-1}\mathbf{W}^{st} + \mathbf{b}^t) \\
\mathbf{W}^{xt}, \mathbf{W}^x &\in \mathbb{R}^{d_{\mathrm{in}} \times d_{\mathrm{hid}}} \\
\mathbf{W}^{st}, \mathbf{W}^s &\in \mathbb{R}^{d_{\mathrm{hid}} \times d_{\mathrm{hid}}} \\
\mathbf{b}^t, \mathbf{b} &\in \mathbb{R}^{1 \times d_{\mathrm{hid}}}
\end{aligned}
$$

Dynamic Connections for RNN



Final Idea: Stopping flow

- For many tasks, it is useful to halt propagation.
- Can do this by applying a reset/forget gate.

$$
\begin{aligned}
\tilde{\mathbf{h}} &= \tanh(\mathbf{x}\mathbf{W}^{x} + (\mathbf{r} \odot \mathbf{s}_{i-1})\mathbf{W}^{s} + \mathbf{b}) \\
\mathbf{r} &= \sigma(\mathbf{x}\mathbf{W}^{xr} + \mathbf{s}_{i-1}\mathbf{W}^{sr} + \mathbf{b}^{r})
\end{aligned}
$$

- Example: Language Modeling

Gated Recurrent Unit (GRU) (Cho et al 2014)

$$
\begin{aligned}
R(\mathbf{s}_{i-1}, \mathbf{x}_i) &= (1 - \mathbf{t}) \odot \tilde{\mathbf{h}} + \mathbf{t} \odot \mathbf{s}_{i-1} \\
\tilde{\mathbf{h}} &= \tanh(\mathbf{x}\mathbf{W}^x + (\mathbf{r} \odot \mathbf{s}_{i-1})\mathbf{W}^s + \mathbf{b}) \\
\mathbf{r} &= \sigma(\mathbf{x}\mathbf{W}^{xr} + \mathbf{s}_{i-1}\mathbf{W}^{sr} + \mathbf{b}^r) \\
\mathbf{t} &= \sigma(\mathbf{x}\mathbf{W}^{xt} + \mathbf{s}_{i-1}\mathbf{W}^{st} + \mathbf{b}^t) \\
\mathbf{W}^{xt}, \mathbf{W}^{xr}, \mathbf{W}^x &\in \mathbb{R}^{d_{\text{in}} \times d_{\text{hid}}} \\
\mathbf{W}^{st}, \mathbf{W}^{sr}, \mathbf{W}^s &\in \mathbb{R}^{d_{\text{hid}} \times d_{\text{hid}}} \\
\mathbf{b}^t, \mathbf{b} &\in \mathbb{R}^{1 \times d_{\text{hid}}}
\end{aligned}
$$

- **t**; dynamic skip-connections
- **r**; reset gating
- **s**; hidden state

Contents

LSTM

LSTMs Development

$$
\begin{aligned}
R(\mathbf{s}_{i-1}, \mathbf{x}_i) &= [\mathbf{c}_i, \mathbf{h}_i] \\
\mathbf{h}_i &= \tanh(\mathbf{c}_i) \\
\mathbf{c}_i &= (1 - \mathbf{t}) \odot \tilde{\mathbf{h}} + \mathbf{t} \odot \mathbf{c}_{i-1} \\
\tilde{\mathbf{h}} &= \tanh(\mathbf{x}\mathbf{W}^{xi} + \mathbf{h}_{i-1}\mathbf{W}^{hi} + \mathbf{b}^{i}) \\
\mathbf{t} &= \sigma(\mathbf{x}\mathbf{W}^{xt} + \mathbf{h}_{i-1}\mathbf{W}^{ht} + \mathbf{b}^{t})
\end{aligned}
$$

The state $\mathbf{s}_i$ is made of 2 components :

- $\mathbf{c}_i$; cell
- $\mathbf{h}_i$; hidden

LSTM Development: Input and Forget Gates

$$
\begin{aligned}
R(\mathbf{c}_{i-1}, \mathbf{x}_i) &= [\mathbf{c}_i, \mathbf{h}_i] \\
\mathbf{h}_i &= \tanh(\mathbf{c}_i) \\
\mathbf{c}_i &= \mathbf{j} \odot \tilde{\mathbf{h}} + \mathbf{f} \odot \mathbf{c}_{i-1} \\
\tilde{\mathbf{h}} &= \tanh(\mathbf{x}\mathbf{W}^{xi} + \mathbf{h}_{i-1}\mathbf{W}^{hi} + \mathbf{b}^{i}) \\
\mathbf{j} &= \sigma(\mathbf{x}\mathbf{W}^{xj} + \mathbf{h}_{i-1}\mathbf{W}^{hj} + \mathbf{b}^{j}) \\
\mathbf{f} &= \sigma(\mathbf{x}\mathbf{W}^{xf} + \mathbf{h}_{i-1}\mathbf{W}^{hf} + \mathbf{b}^{f})
\end{aligned}
$$

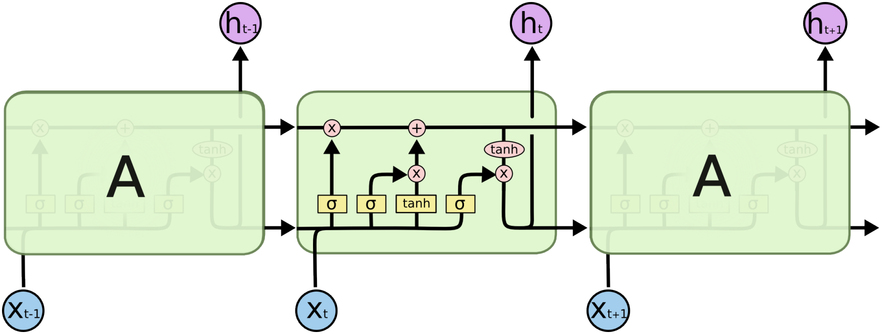
No longer a convex combination.

- $\mathbf{c}_i$; cell
- $\mathbf{h}_i$; hidden
- $\mathbf{j}$; input gate
- $\mathbf{f}$; forget gate

Long Short-Term Memory

$$
\begin{aligned}
R(\mathbf{s}_{i-1}, \mathbf{x}_i) &= [\mathbf{c}_i, \mathbf{h}_i] \\
\mathbf{c}_i &= \mathbf{j} \odot \mathbf{i} + \mathbf{f} \odot \mathbf{c}_{i-1} \\
\mathbf{h}_i &= \tanh(\mathbf{c}_i) \odot \mathbf{o} \\
\mathbf{i} &= \tanh(\mathbf{x}\mathbf{W}^{xi} + \mathbf{h}_{i-1}\mathbf{W}^{hi} + \mathbf{b}^i) \\
\mathbf{j} &= \sigma(\mathbf{x}\mathbf{W}^{xj} + \mathbf{h}_{i-1}\mathbf{W}^{hj} + \mathbf{b}^j) \\
\mathbf{f} &= \sigma(\mathbf{x}\mathbf{W}^{xf} + \mathbf{h}_{i-1}\mathbf{W}^{hf} + \mathbf{b}^f) \\
\mathbf{o} &= \tanh(\mathbf{x}\mathbf{W}^{xo} + \mathbf{h}_{i-1}\mathbf{W}^{ho} + \mathbf{b}^o)
\end{aligned}
$$

- **f**; forget gate
- **i**; input gate



ht-1
ht
ht+1
A
σ
σ
tanh
σ
tanh
A
Xt-1
Xt
Xt+1