Long Short-Term Memory (LSTM)

M1 Yuichiro Sawai
Computational Linguistics Lab.

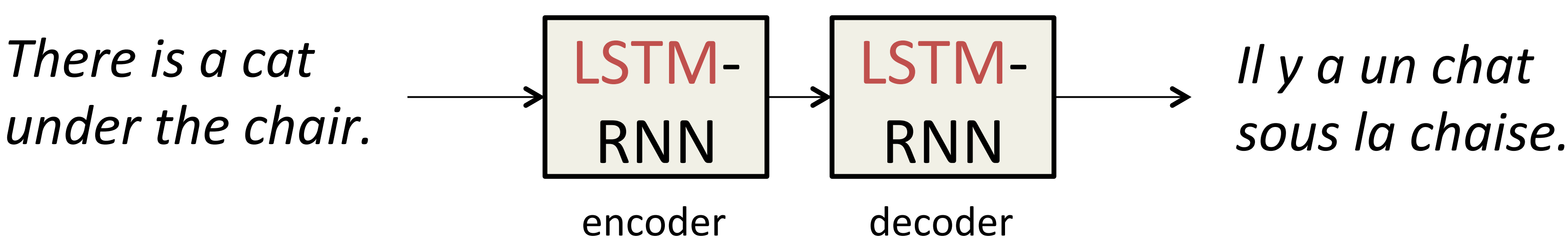
January 15, 2015 @ Deep Lunch

Why LSTM?

- Often used in many recent RNN-based systems
  - Machine translation
  - Program execution
- Can capture “long-term dependency”

Sequence to Sequence Learning with Neural Networks [Sutskever+14]

- Machine translation (English to French)
- Achieved state-of-the-art, while making little assumption about data



Learning to Execute [Zaremba+14]

- Trained an RNN to execute a code written in Python-like language

```
j=8584
for x in range(8):
   j+=920
b=(1500+j)
print((b+7567))
```

25011.

Review: Feed-Forward Neural Network

output layer

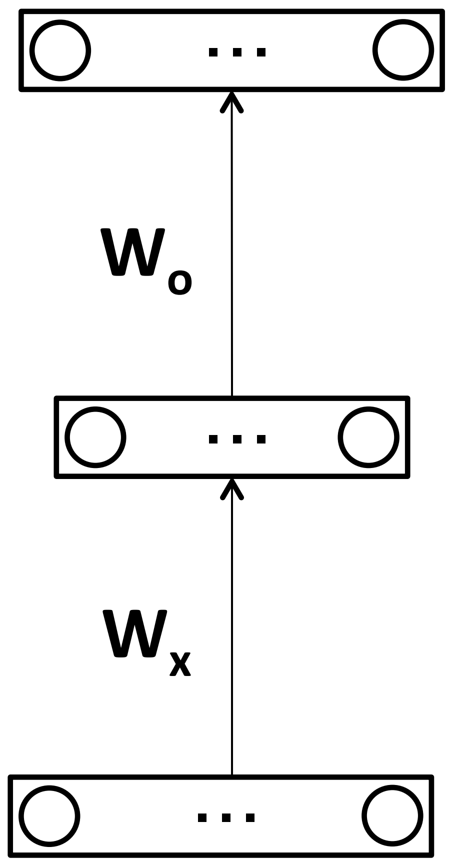
OUT: $\mathbf{y}=\exp(\mathbf{W}_o\mathbf{h})/Z$

hidden layer

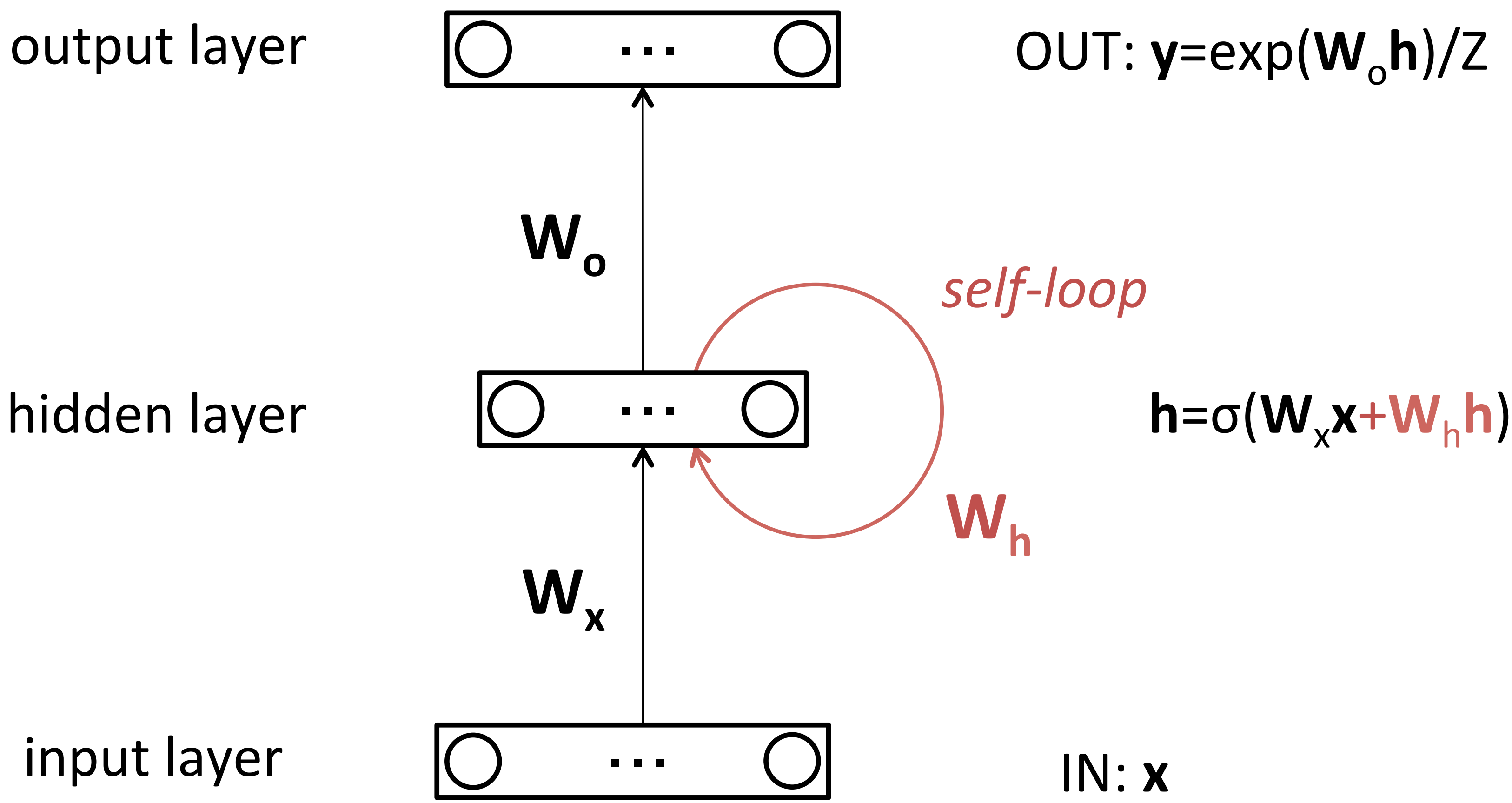
$\mathbf{h}=\sigma(\mathbf{W}_x\mathbf{x})$

input layer

IN: $\mathbf{x}$

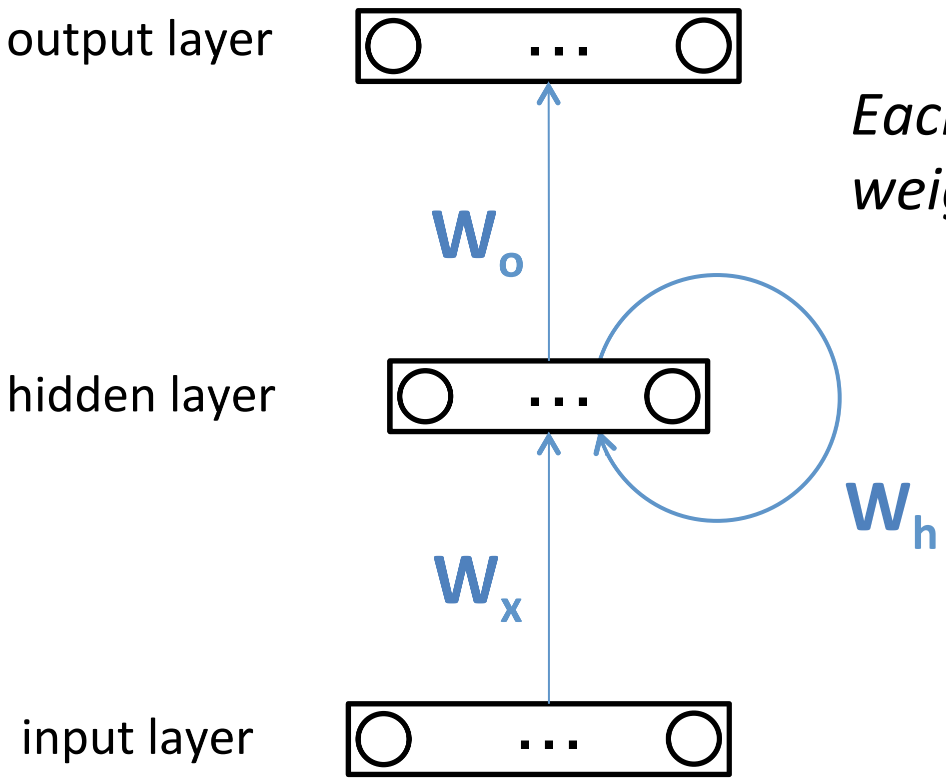


Review: Recurrent Neural Network



Review: Training of RNN

Find “good” values for W_x, W_h, W_o



Each arrow has a corresponding weight that has to be optimized.

Review: Gradient Descent

Iteratively update weight by moving along gradient

$$W_{new} = W_{old} + \alpha \frac{\partial E}{\partial W}$$

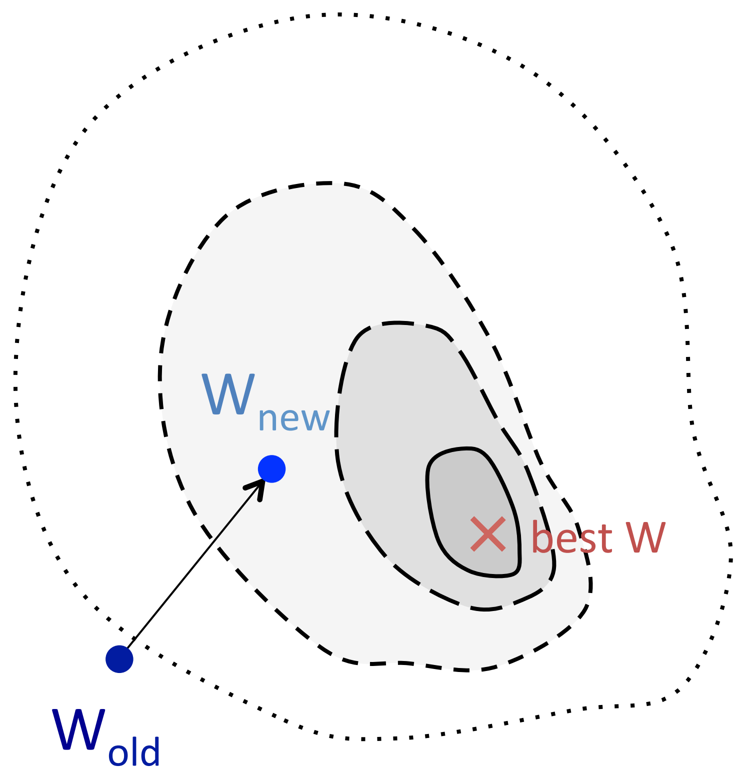
E: error function

(cross-entropy for multi-classification)

α: learning rate

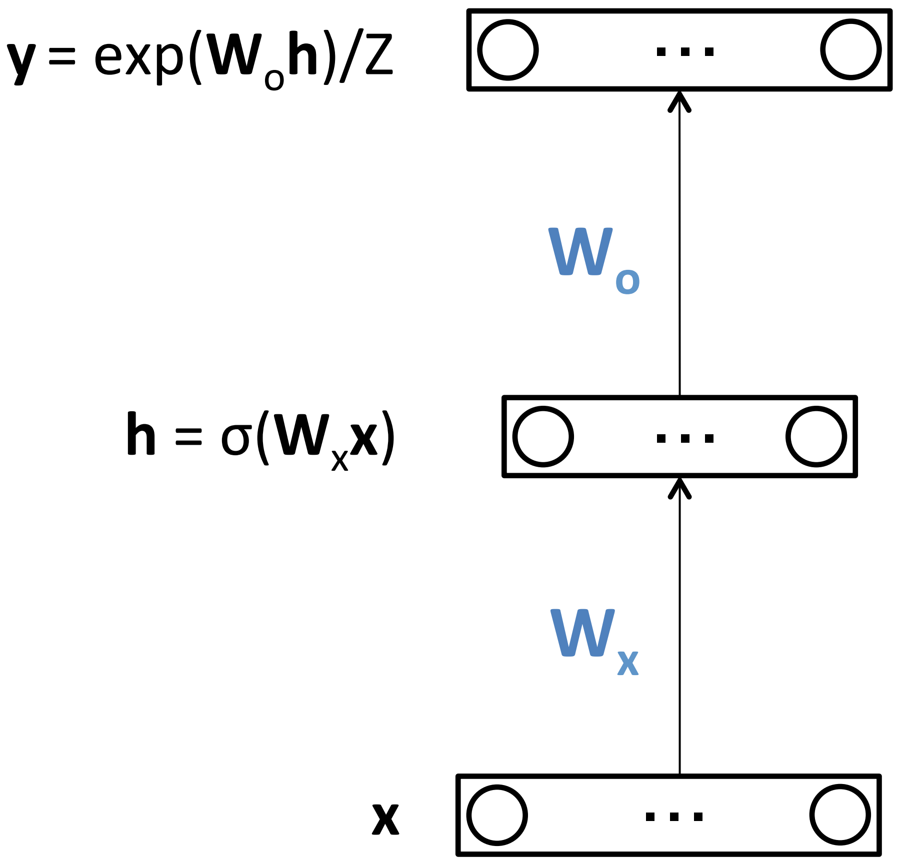
How is $\frac{\partial E}{\partial W}$ calculated?

Back-propagation!



Review: Gradient Calculation

Gradients are calculated from errors(δ) propagated backward.



$\delta_o = y - y_{correct}$

$$\frac{\partial E}{\partial W_o} = \delta_o h^T$$

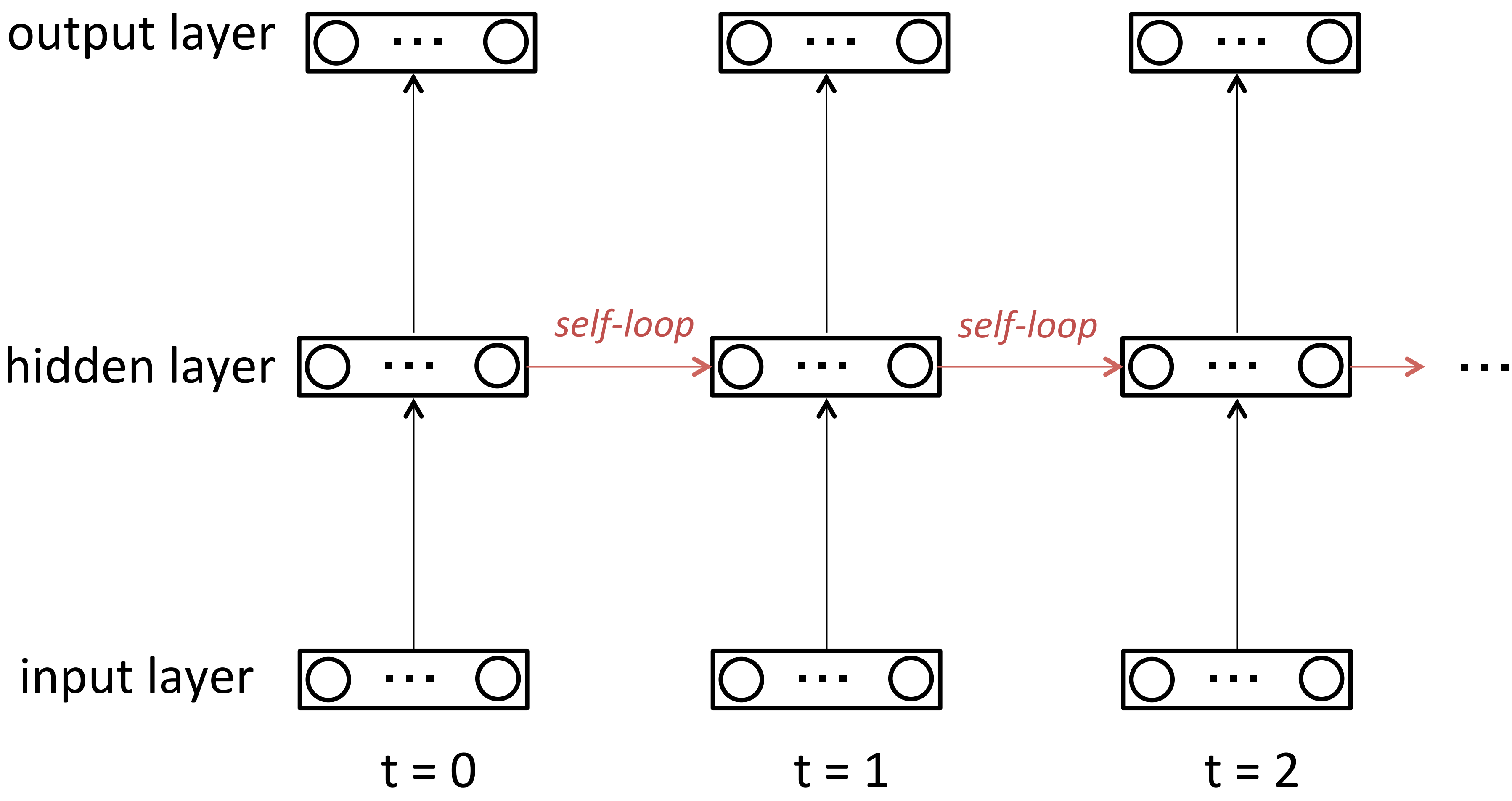
$\delta_h = \sigma'(\mathbf{W}_x\mathbf{x}) \circ \mathbf{W}_o^T\delta_o$

$$\frac{\partial E}{\partial W_x} = \delta_h x^T$$

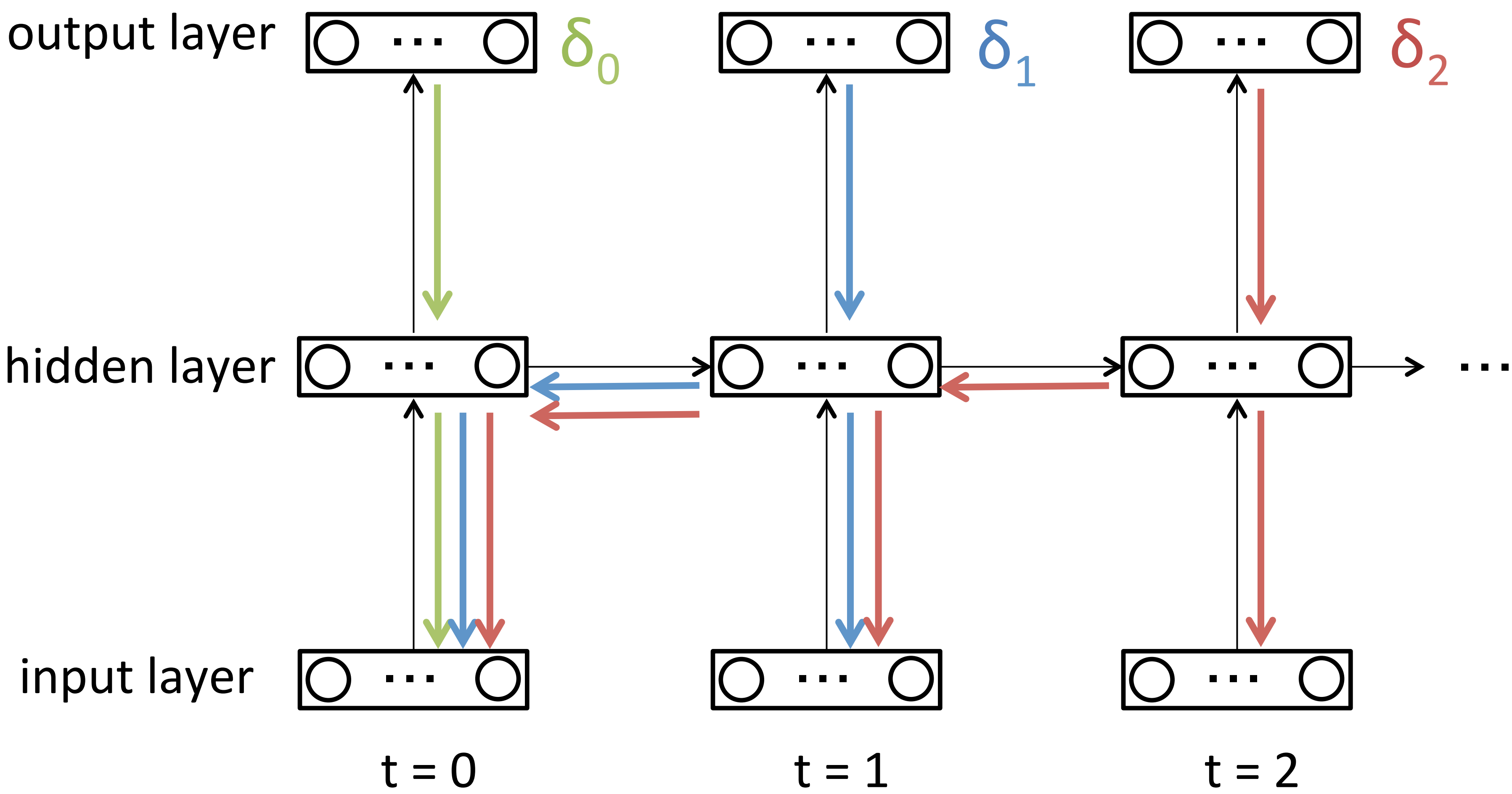
Error gets multiplied by the weight.

backward propagation of error

Review: Unfolding RNN through Time

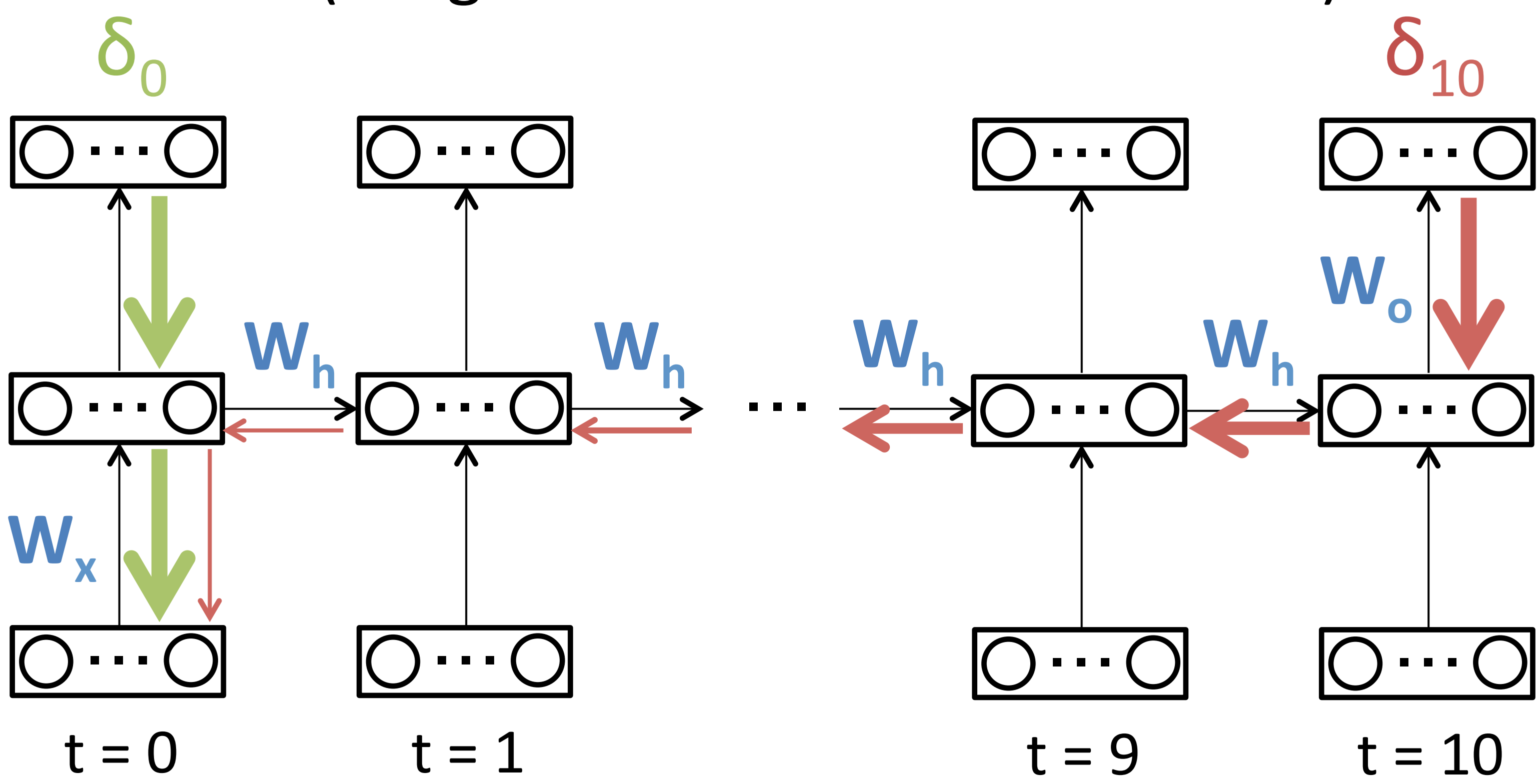


Review: Back Propagation through Time (BPTT)



Problem: Vanishing Gradients

Remember: error is multiplied by weight each time
(weights are often smaller than 1)



Errors (thus gradients) gets smaller exponentially!

Why does it Matter?

- Cannot learn “long-term dependency”
- “Long-term dependency” emerges when input signal and teacher signal are far apart (>10 time steps).

Long-Term Dependency

- You are presented with the following 6 sequences consisting of `A`s, `B`s, `X`s.

1: `AXXXXXXXXXXXA`
2: `AXXXXXXXXXXXA`
3: `BXXXXXXXXXXXB`
4: `AXXXXXXXXXXXA`
5: `BXXXXXXXXXXXB`
6: `AXXXXXXXXXXXA`

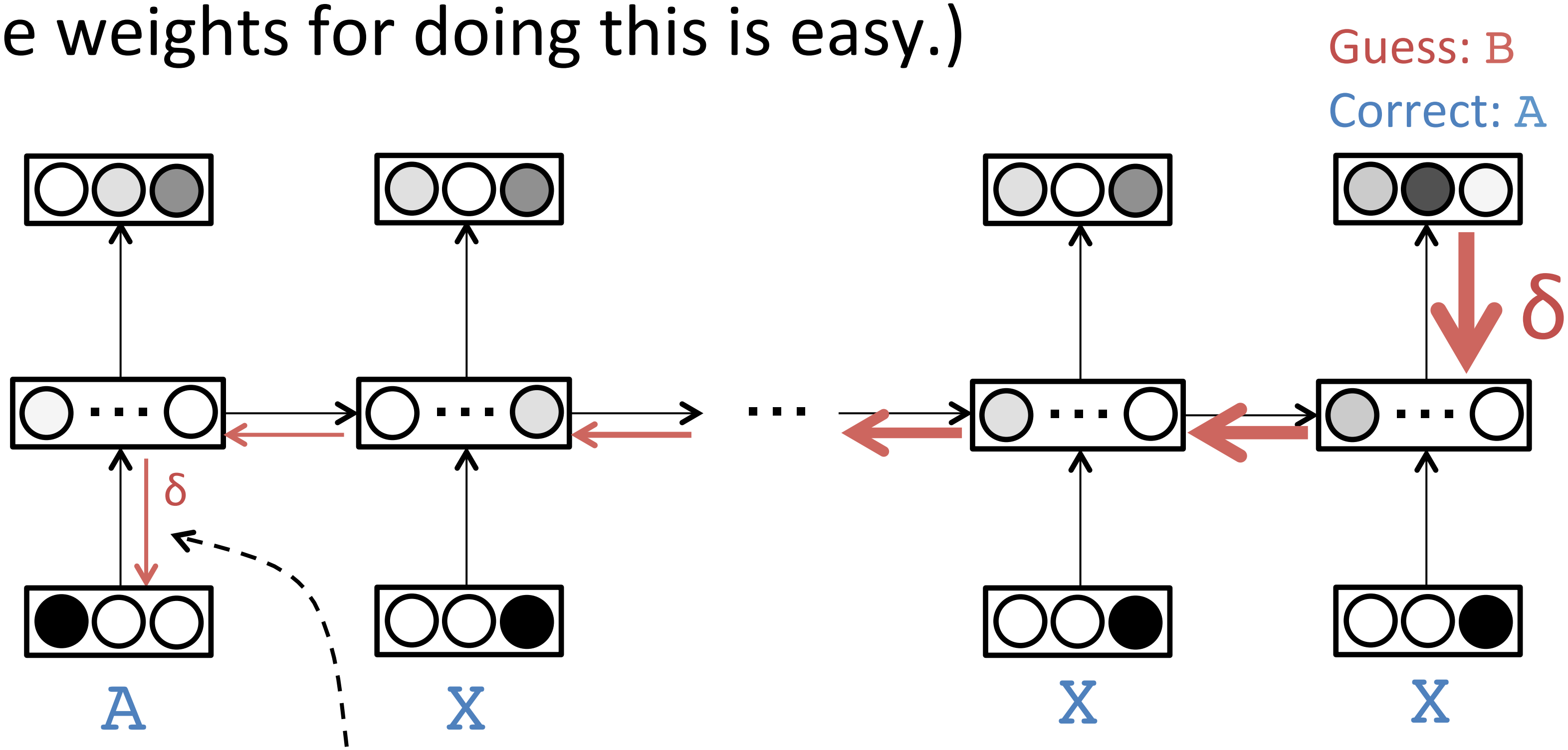
- Can you guess what comes next?

`AXXXXXXXXXXX` $\boxed{?}$

Simplified version of Task 2a from [Hochreiter+97]

Unfortunately, RNN with BPTT fails.

RNN must find out **by itself** that it must store the first letter in the hidden layer. (Note that manually setting the weights for doing this is easy.)



Only place where RNN could reprogram itself (update weights) to store the first letter in the hidden layer, but error is too small!

Solutions

- Alternative training techniques
  - Hessian-free optimization [Martens+11]
- Alternative architectures (training remains the same)
  - **Long Short-term Memory (LSTM)**

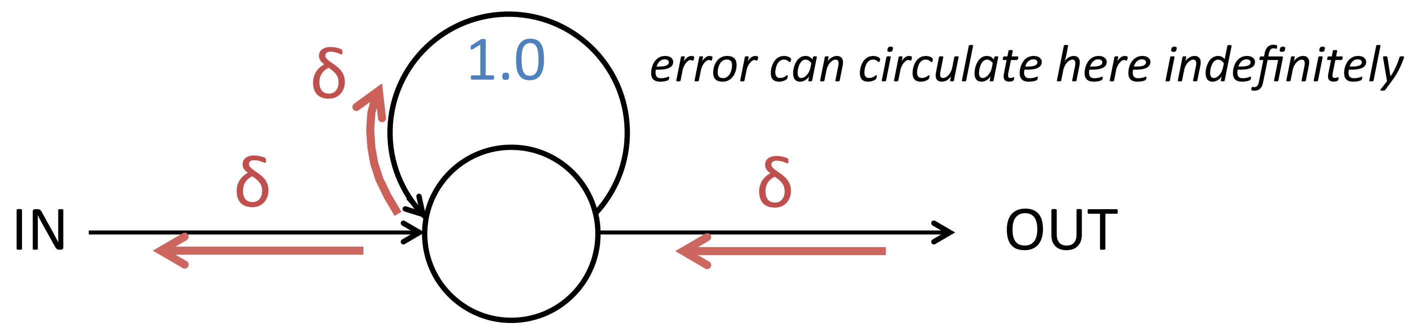
Why does Gradient Vanish?

- Because error(δ) is multiplied by scaling factors.

  $\delta_i = \sigma'(y_i) W_{ji}\, \delta_j$ (connection from unit i to unit j)

- What if we set $W_{ii} = 1$ and $\sigma(x) = x$?

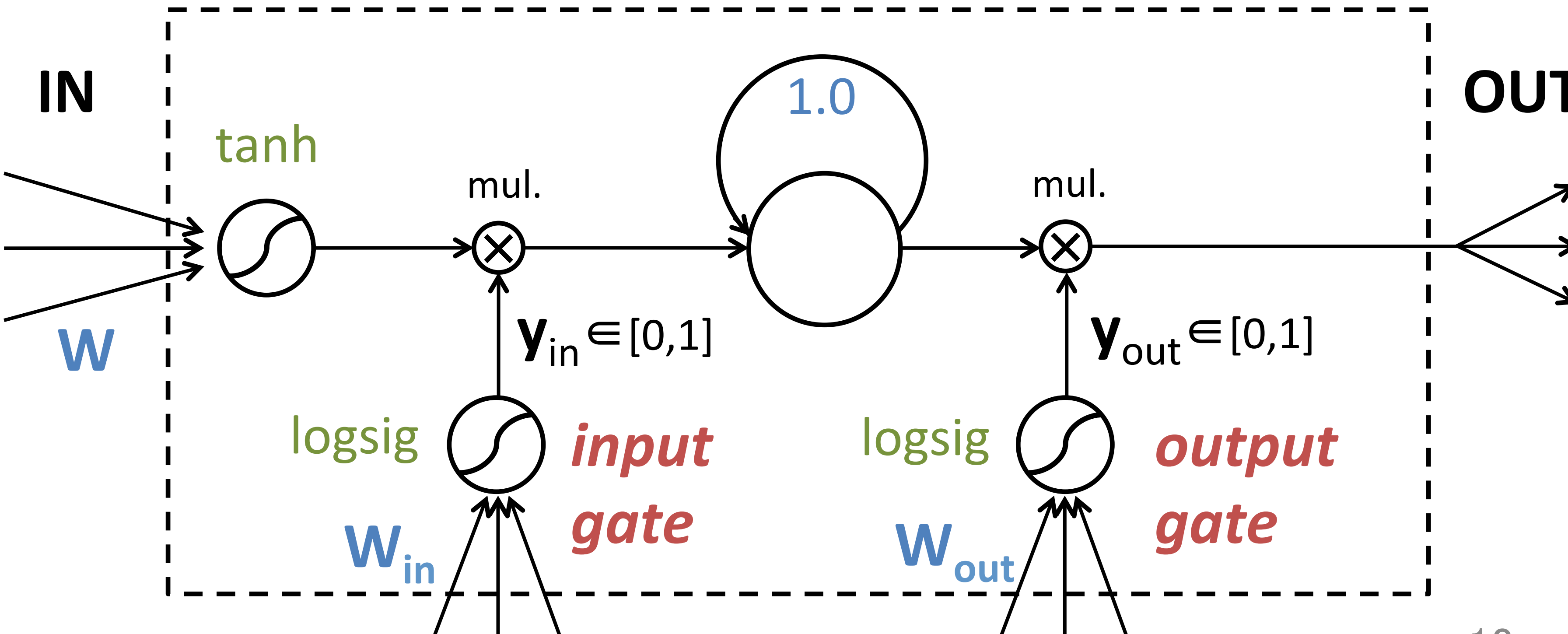
  → Constant Error Carrousel (CEC)



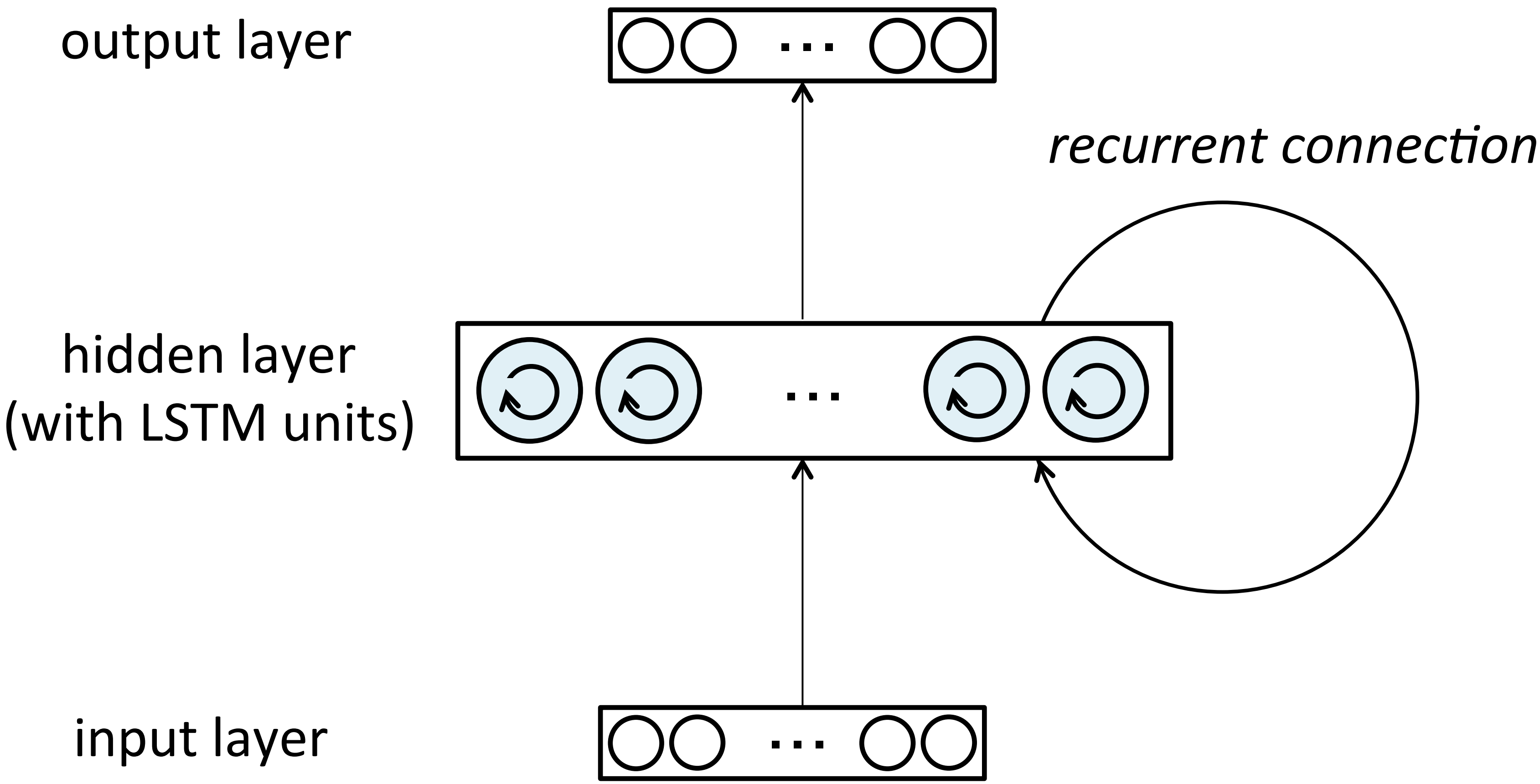
- Constant Error Carrousel (CEC) enables error propagation of arbitrary time steps.

LSTM Unit

- CEC with “input gate” and “output gate” is LSTM unit
- Gates are for controlling error flow depending on the context.



Entire Picutre of LSTM-RNN



LSTM Variants

- Forget gates
  - explicitly reset the value in CEC
- Peephole connections
- Multiple CECs in an LSTM unit
- etc.

- Common: CEC and gates

Summary

- RNN with BPTT fails to learn “long-term dependency” due to vanishing gradients.
- LSTM overcomes this problem by having Constant Error Carrousel (CEC).
- LSTM has input and output gates.

References

- Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with LSTM recurrent networks. The Journal of Machine Learning Research, Vol. 3, pp. 115–143, 2003.
- Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, Vol. 1. IEEE Press, 2001.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, Vol. 9, No. 8, pp. 1735–1780, 1997.
- James Martens and Ilya Sutskever. Learning recurrent neural networks with Hessian-free optimization. In Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 1033–1040, 2011.
- Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. LSTM neural networks for language modeling. In INTERSPEECH, pp. 194–197, 2012.
- Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems, pp. 3104–3112, 2014.
- Wojciech Zaremba and Ilya Sutskever. Learning to execute. arXiv preprint arXiv:1410.4615, 2014.