PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

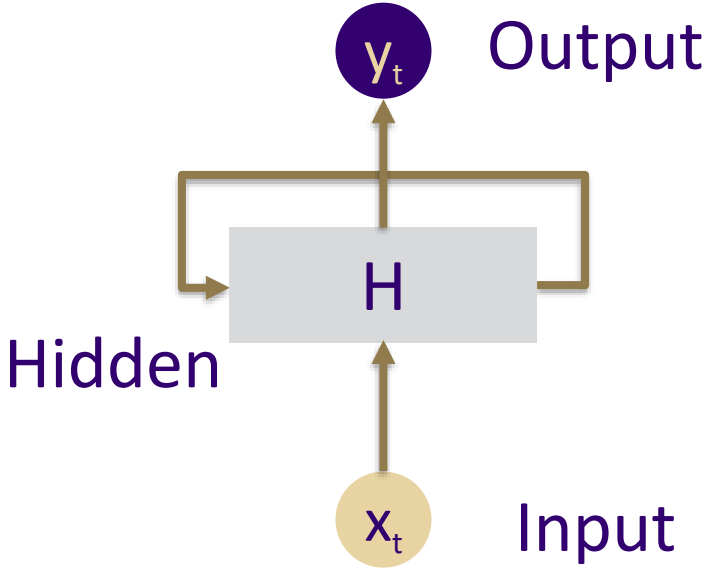
Machine Learning Techniques

DATASCI 420

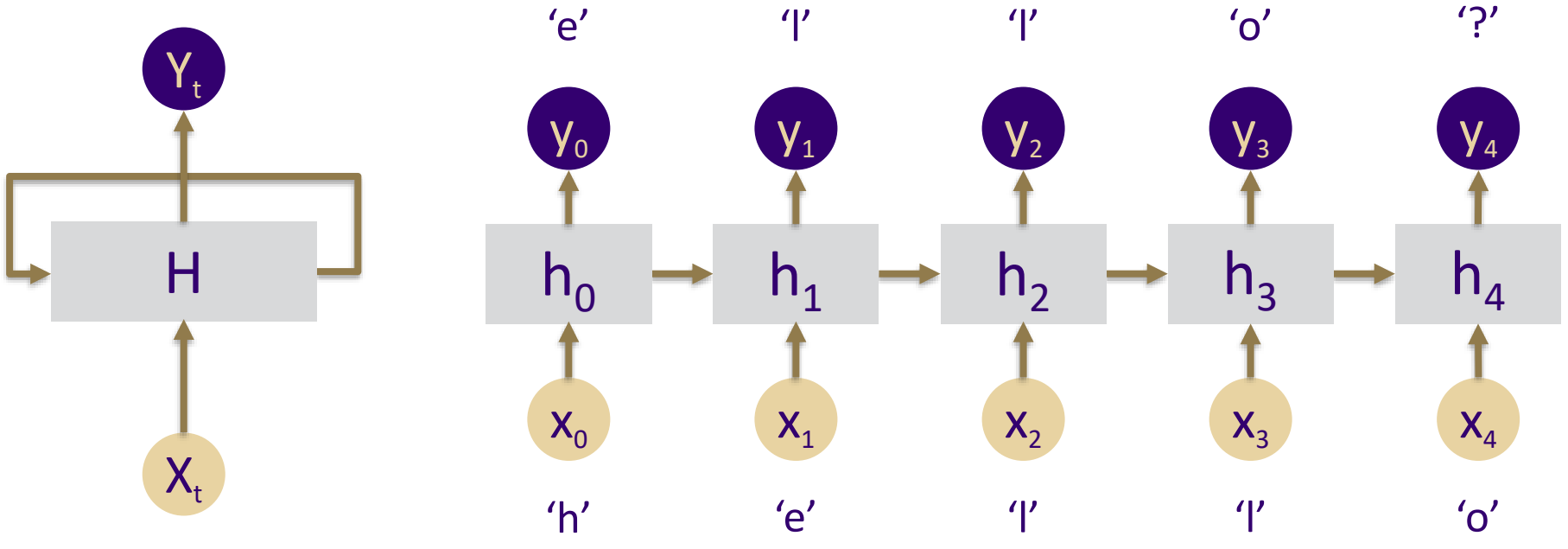
Lesson 10 Deep Learning II: Recurrent Neural Networks

What are RNNs

- ANN for sequential or times series data
- Perform the same task for every element of a sequence



Unfolding the RNN



"Vanilla" RNN Cell

- Model Parameters:

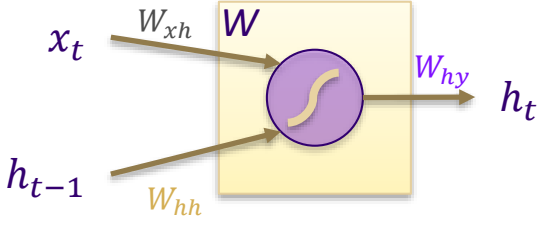
$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$y_t = W_{hy} h_t$$

Weight: hidden to hidden

Weight: hidden to hidden

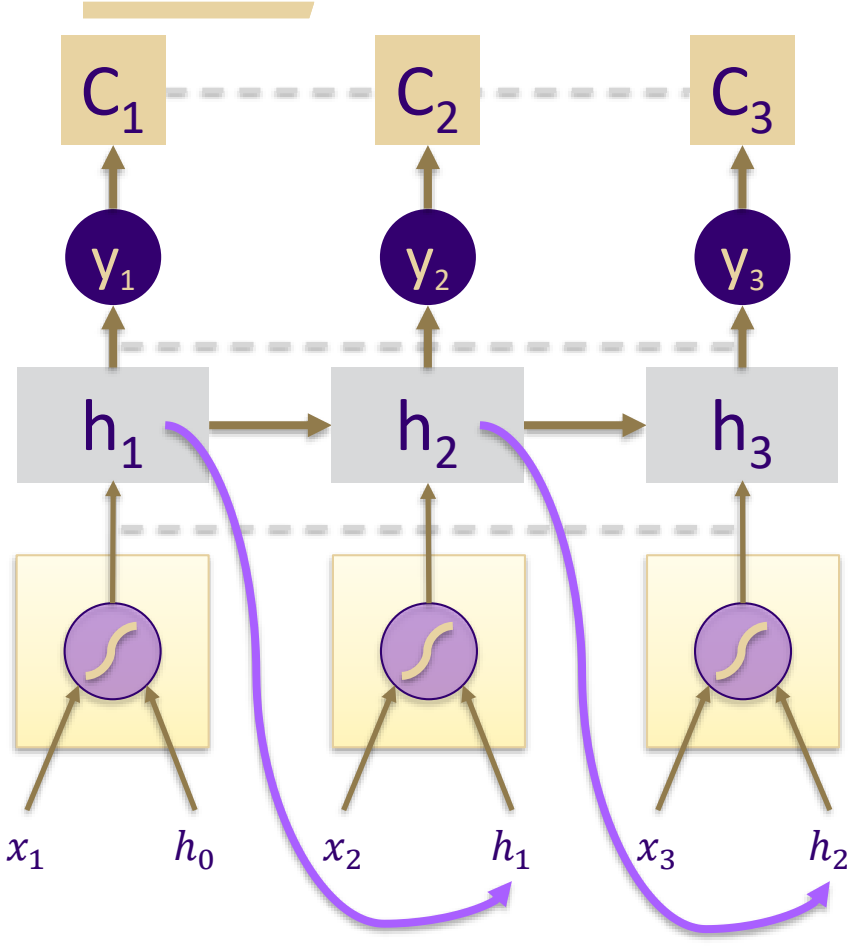
Weight: input to hidden



$$P(y_t | x_1, \ldots, x_t)$$

```
class RNN:
 def step(self, x):
    self.h = np.tanh(np.dot(self.W_hh, self.h) +
             np.dot(self.W_xh, x))
    y = np.dot(self.W_hy, self.h)
    return y
```

RNN Forward



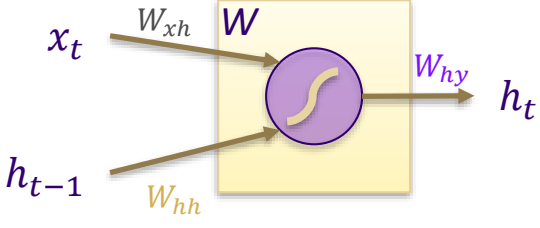
$>y_t = F(h_t)$

$>C_t = Loss(y_t GT_t)$

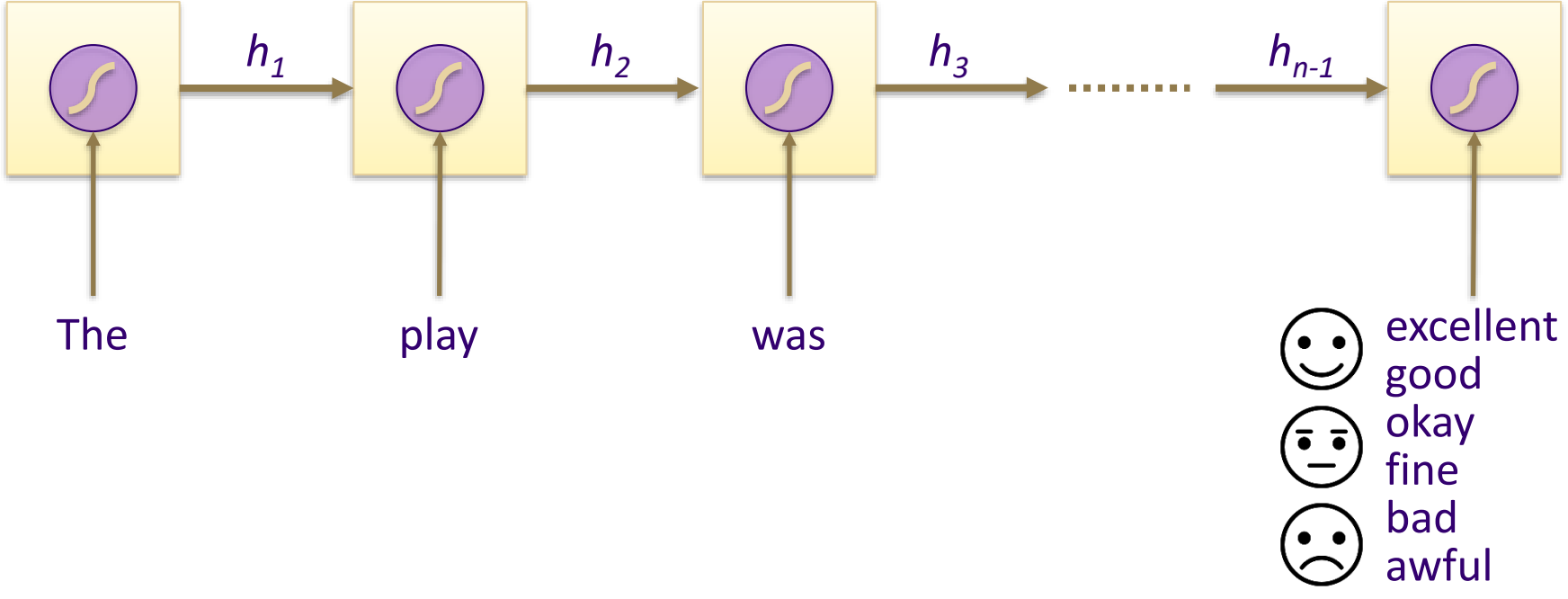
Shared Weights

RNN Ideas

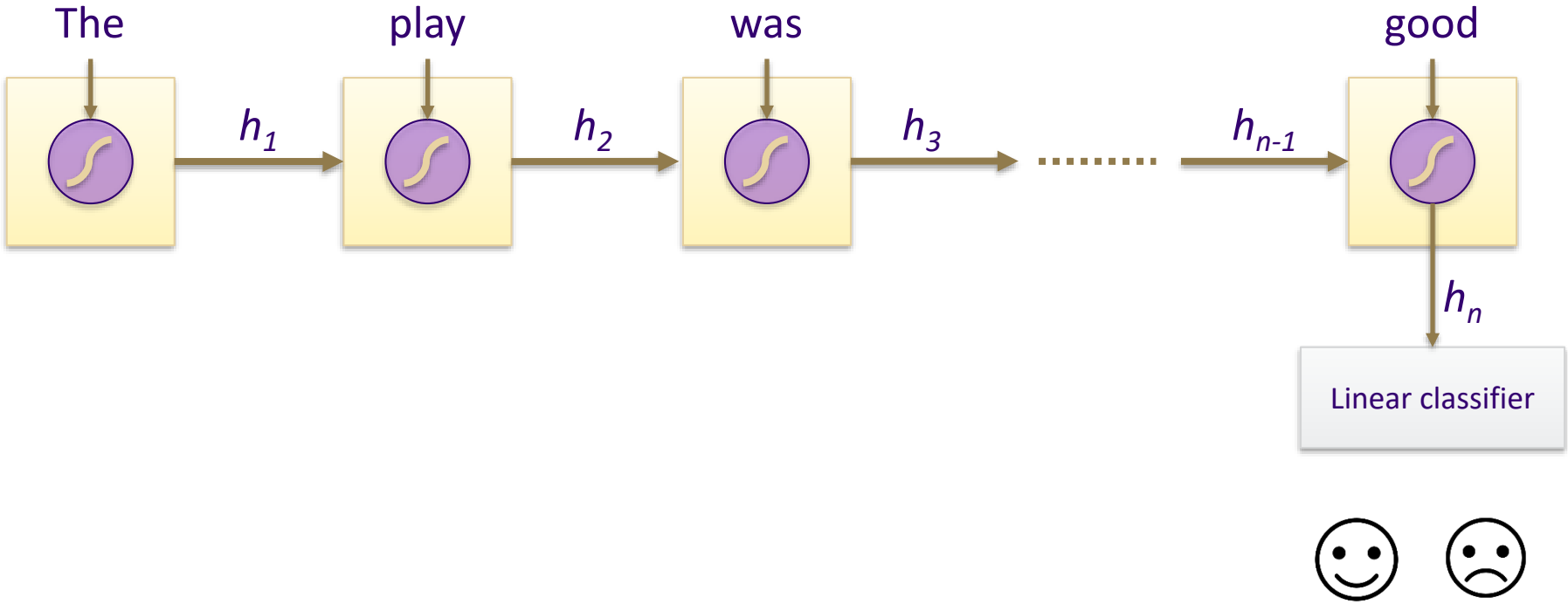
- Weights are shared over time
- Copies of the RNN cells roll out over time given differing inputs and time steps

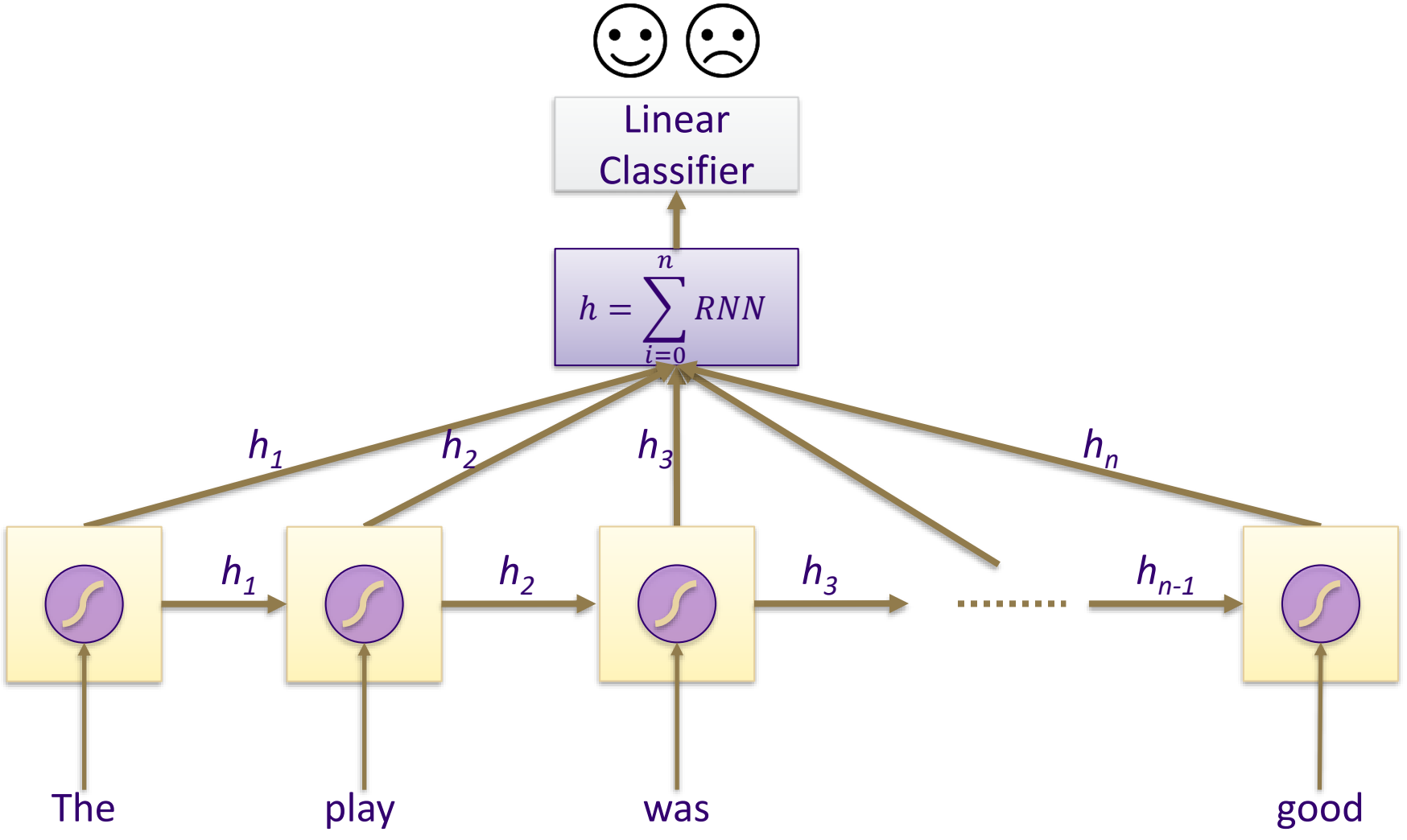


Word-based Example: Sentiment Classification



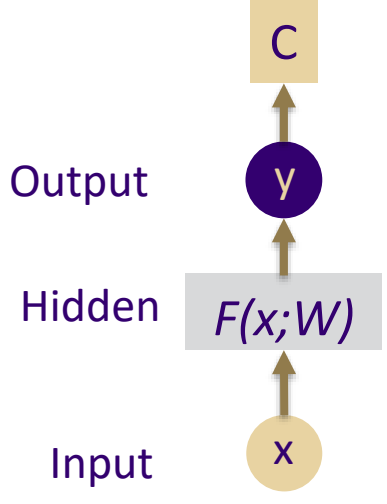
Word-based Example: Sentiment Classification




Linear Classifier
$h = \sum_{i=0}^{n} RNN$
h_1
h_2
h_3
h_n
h_1
h_2
h_3
h_{n-1}
The
play
was
good

Standard Backpropagation and SDG

Finding the Derivatives of cost with respect to any variable



- $y = f(x; W)$
- $C = loss(y, y_{GT})$

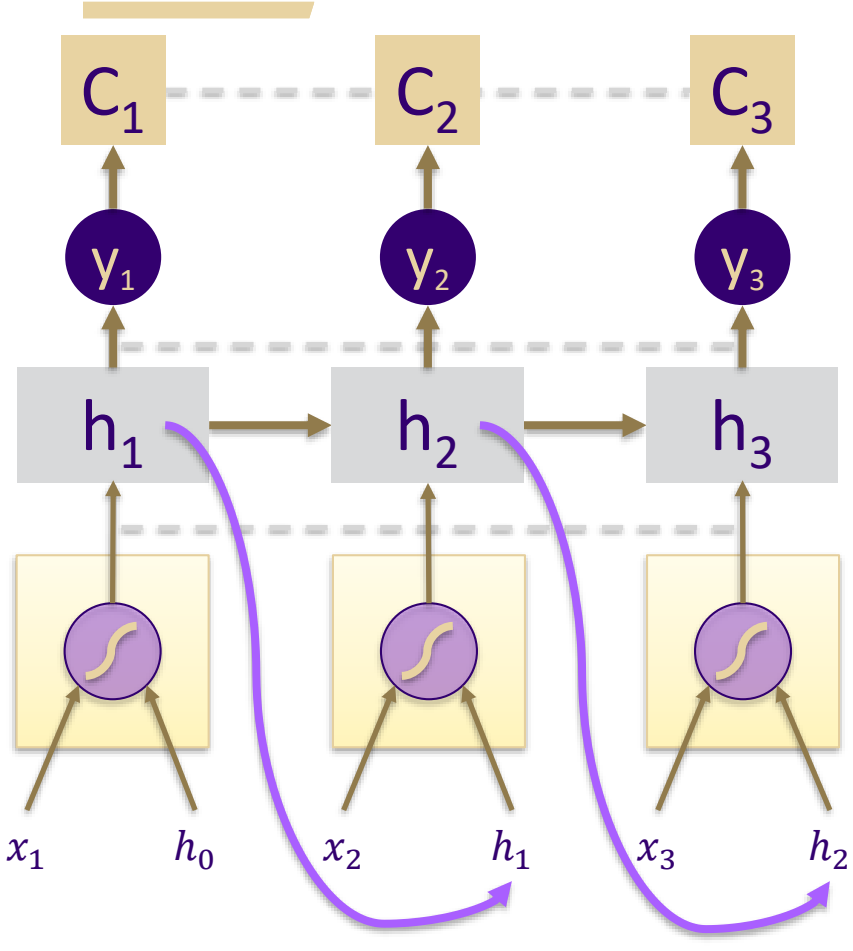
SGD

- $W \leftarrow W - \eta \frac{\partial C}{\partial W}$
- $\frac{\partial C}{\partial W} = \left(\frac{\partial C}{\partial y}\right)\left(\frac{\partial y}{\partial W}\right)$

Backpropagation Through Time (BPTT)

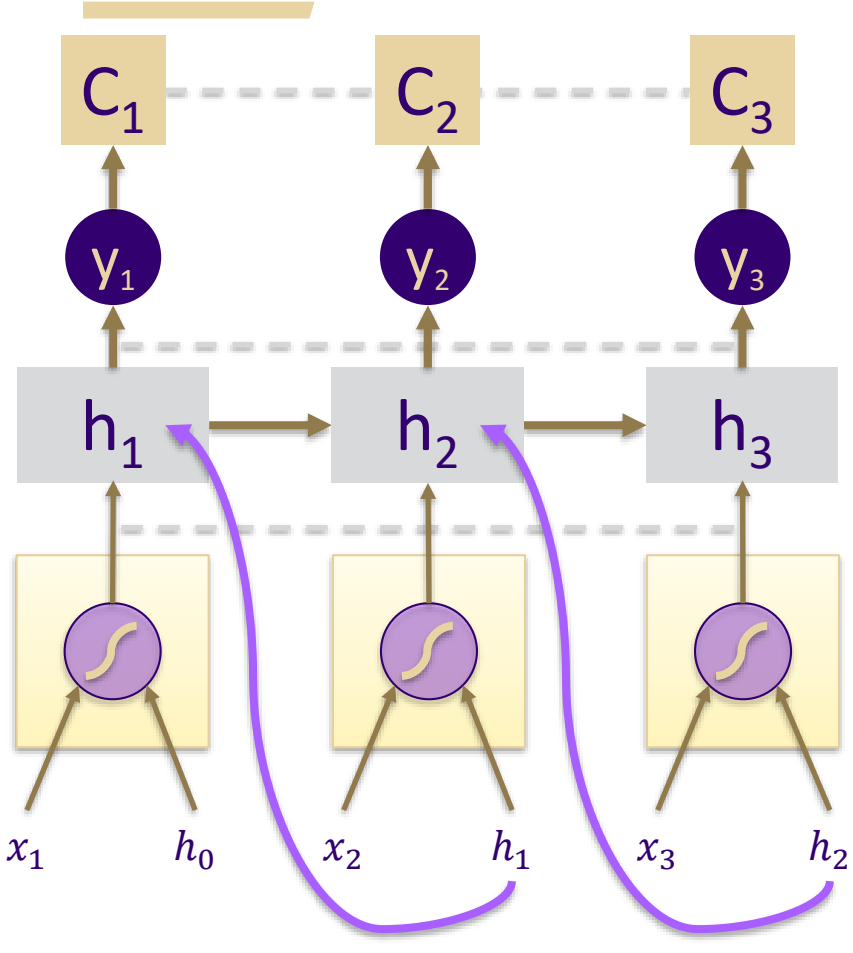
- One of the methods used to train RNNs
- Accepts as an input the entire time series from the (unfolded) generated on the feed-forward pass
- Weights updates are computed for each RNN cell in the unfolded network, which are summed (or averaged) and then applied to the RNN weights

RNN Forward



- Treat the unfolded network as a big FF ANN
- Take the entire sequence as input
- Compute the gradients through BP/SDG
- Update the shared weights

RNN Backwards



- Treat the unfolded network as a big FF ANN
- Take the entire sequence as input
- Compute the gradients through BP/SDG
- Update the shared weights

Challenges of RNNs

- Vanishing or exploding gradient problems

  >Can use thresholds or normalized clipping
- ReLu for RNNs: https://arxiv.org/abs/1504.00941
- Long Short-Term Memory (LSTM)

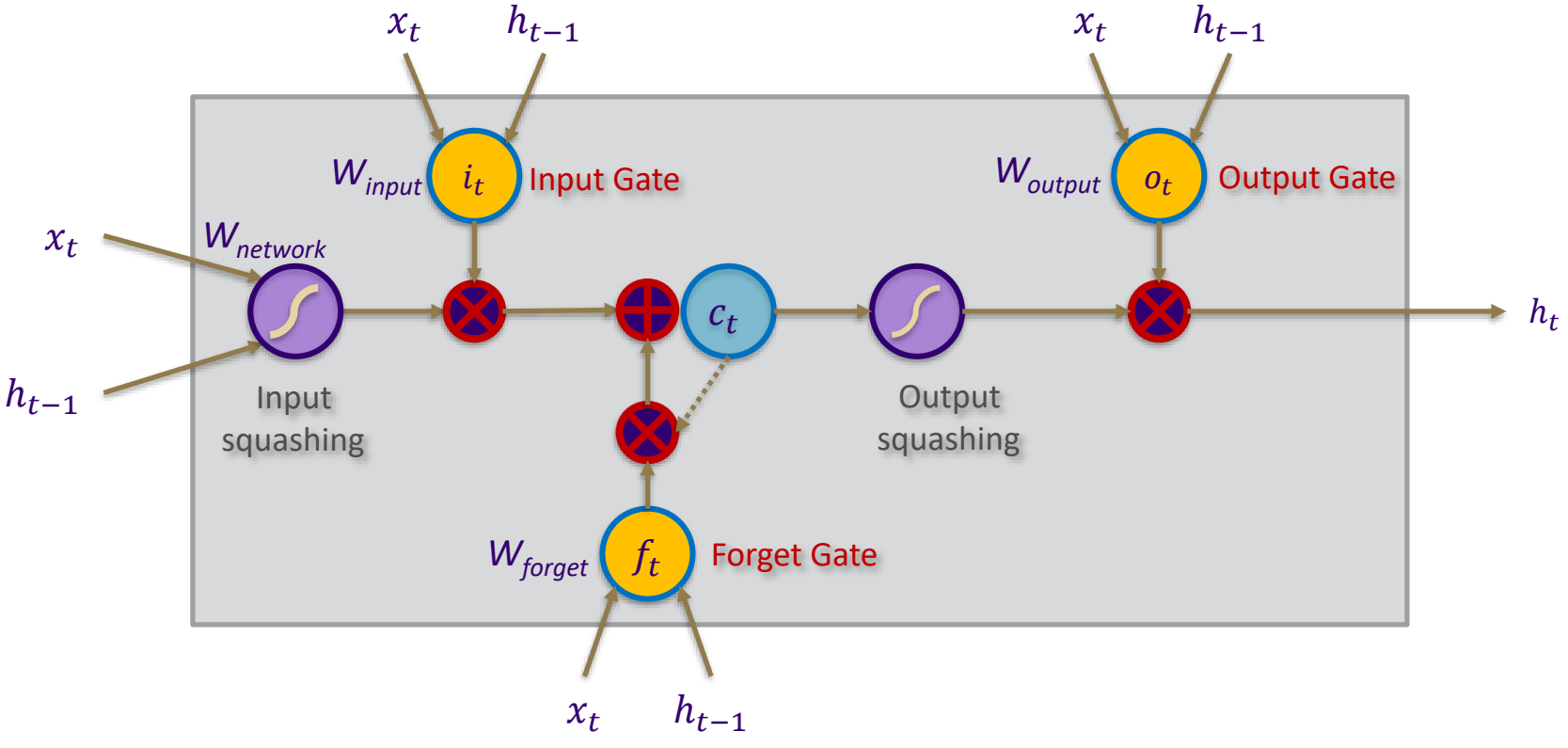
Long Short Term Memory Networks (LSTM)

- Preserve error backpropagated through time and layers
- Enable the network to continue to learn over longer time steps
- Creates the ability to links causes and effects that are more remote from one another
- Works well when reinforcing signals are sparse or delayed

Components of LSTM

- Gate: Optionally let information through
- Cell state: long term memory
- Forget gate: determine what old information to forget
- Input gate: determine what new information to store
- Output gate: decide what to output

LSTM Cell


x_t
h_{t-1}
W_{input}
i_t
Input Gate
x_t
h_{t-1}
W_{output}
o_t
Output Gate
x_t
$W_{network}$
h_{t-1}
c_t
h_t
Input
squashing
Output
squashing
W_{forget}
f_t
Forget Gate
x_t
h_{t-1}

RNN Hyperparameter Tuning

- Learning rate is the single most important hyperparameter
- Normalize the data
- Try different initialization of the weight
- Evaluate test set performance at each epoch to know when to stop
- Alter the number of hidden layers
- Try several activation functions, softsign (not softmax) is an alternative to tanh
- Momentum applies and you can also add other types of updaters (RMSProp and AdaGrad)
- Watch out for overfitting
  - > You can use regularization to help with overfitting: such as L1 and L2 normalization and dropout among others
  - > The larger the network, the more powerful, but it's also easier to overfit→Feature optimization is important
  - > More data is almost always better, because it helps fight overfitting
  - > Tune the number of epochs (complete passes through the dataset)

A Great Resource:

- Andrej Karpathy's Blog:
- http://karpathy.github.io/2015/05/21/rnn-effectiveness/