Recurrent Neural Networks

Eilif Solberg

TEK5040/TEK9040

Outline

Introduction

The dimension of time

- Inputs arrive in a sequence
- Actions performed one after another

The dimension of time

- Inputs arrive in a sequence
- Actions performed one after another

Why process data serially?

The dimension of time

- Inputs arrive in a sequence
- Actions performed one after another

Why process data serially?

- Need to respond immediately

The dimension of time

- Inputs arrive in a sequence
- Actions performed one after another

Why process data serially?

- Need to respond immediately
- Limited *bandwidth* for “sensor” inputs

The dimension of time

- Inputs arrive in a sequence
- Actions performed one after another

Why process data serially?

- Need to respond immediately
- Limited *bandwidth* for "sensor" inputs
- Limited *computational* capability

The dimension of time

- Inputs arrive in a sequence
- Actions performed one after another

Why process data serially?

- Need to respond immediately
- Limited *bandwidth* for “sensor” inputs
- Limited *computational* capability
- Limited *storing* capability

The dimension of time

- Inputs arrive in a sequence
- Actions performed one after another

Why process data serially?

- Need to respond immediately
- Limited *bandwidth* for "sensor" inputs
- Limited *computational* capability
- Limited *storing* capability
- More efficient to divide work into subtasks?

How do you process a sentence?

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

How do you process a sentence?

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

- One character at a time?

How do you process a sentence?

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

- One character at a time?
- One word at a time?

How do you process a sentence?

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

- One character at a time?
- One word at a time?
- What if you were new to the language?

How do you process a sentence?

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

- One character at a time?
- One word at a time?
- What if you were new to the language?
- What if all letters where mirrored?

How do you process a sentence?

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

- One character at a time?
- One word at a time?
- What if you were new to the language?
- What if all letters where mirrored?
- Will look at models that combine serial and parallel processing for sequence data

Example applications

Example applications

- Machine translation
- Sentiment analysis
- Time series models
- Image captioning
- Language modeling in general, character and word based
- State representation in reinforcement learning

Categories

- Sequence-to-vector
- Vector-to-sequence
- Sequence-to-sequence
- Sequence-to-sequence of different lengths...

Formal model

- Let $s^t \in \mathbb{R}^d$ represent our *state* at time t
- Let $x^t \in \mathbb{R}^m$ denote the input at time t
- Let $y^t \in \mathbb{R}^n$ denote the output at time t

Formal model

- Let $s^t \in \mathbb{R}^d$ represent our *state* at time t
- Let $x^t \in \mathbb{R}^m$ denote the input at time t
- Let $y^t \in \mathbb{R}^n$ denote the output at time t

In our model we have $y^t = f(s^t)$

Formal model

- Let $s^t \in \mathbb{R}^d$ represent our *state* at time t
- Let $x^t \in \mathbb{R}^m$ denote the input at time t
- Let $y^t \in \mathbb{R}^n$ denote the output at time t

In our model we have $y^t = f(s^t)$
How do we update beliefs and plans?

Formal model

- Let $s^t \in \mathbb{R}^d$ represent our *state* at time t
- Let $x^t \in \mathbb{R}^m$ denote the input at time t
- Let $y^t \in \mathbb{R}^n$ denote the output at time t

In our model we have $y^t = f(s^t)$
How do we update beliefs and plans? Models of the form

$$s^t = h(x^t, s^{t-1}, y^{t-1})$$

RNN I

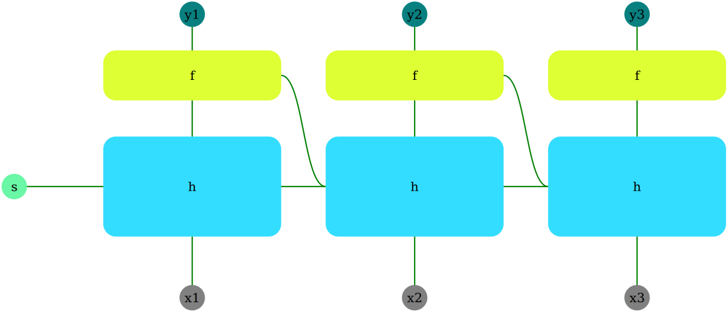


Figure: RNN model with initial state s, unrolled three time steps. The output of f flowing to the next state at time t is the output y^t.

RNN II

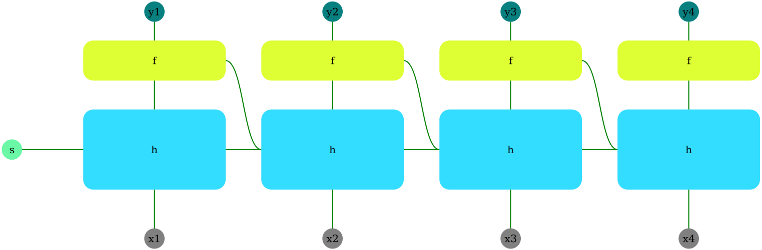


Figure: RNN model, unrolled four time steps

RNN III

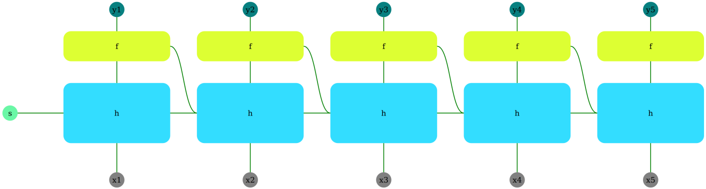
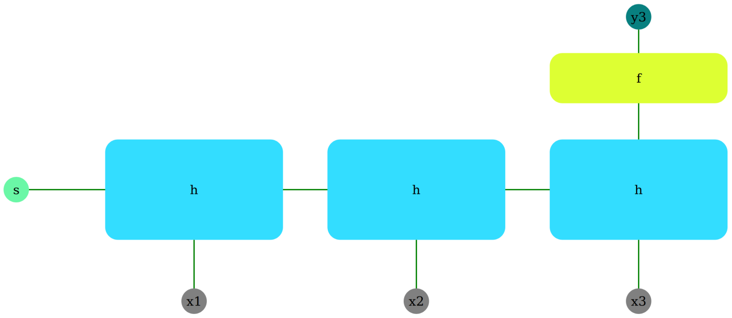
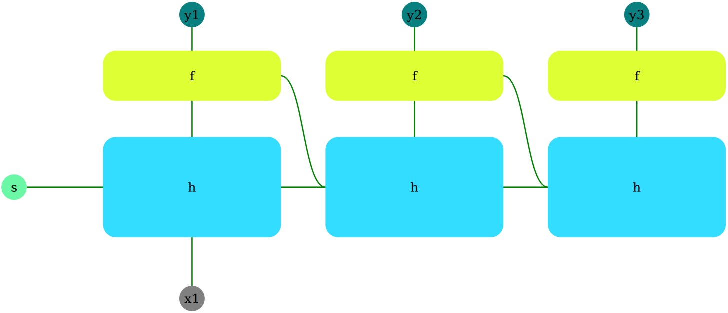


Figure: RNN model, unrolled five time steps

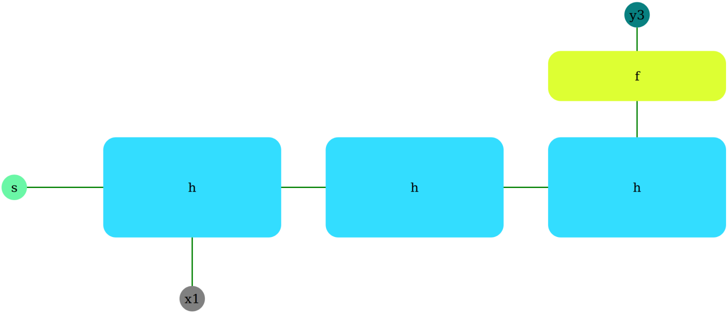
RNN IV - single output



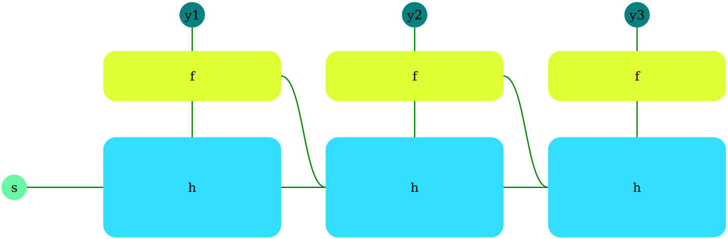
RNN V - single input



RNN V - single input, single output



RNN VI - no input



Vanilla RNN

Model

$$h(x, s, y) = a(Ux + Vs + Wy + b) \tag{1}$$

where a is an activation function and

- $U \in \mathbb{R}^{d\times m}$
- $V \in \mathbb{R}^{d\times d}$
- $W \in \mathbb{R}^{d\times n}$
- $b \in \mathbb{R}^{d}$

Note: Equation (1) equivalent to $a(M[x, s, y] + b)$ where $M = [U, V, W]$.

Vanilla RNN Cell in TensorFlow

```
class VanillaRNNCell(tf.keras.layers.AbstractRNNCell):
    def __init__(self, units):
        super(VanillaRNNCell, self).__init__()
        self.units = units
        self.dense = layers.Dense(units)

    @property
    def state_size(self):
        return self.units

    # input and output already concatenated into 'x' (possibly
    ↪ after preprocessing)
    def call(self, x, state):
        # [batch_size, num_inputs] x [batch_size, units] ==>
        ↪ [batch_size, num_inputs+units]
        c = tf.concat([x, state], axis=-1)
        h = self.dense(c)
        output = activations.tanh(h)
        return output, output
```

Vanilla RNN

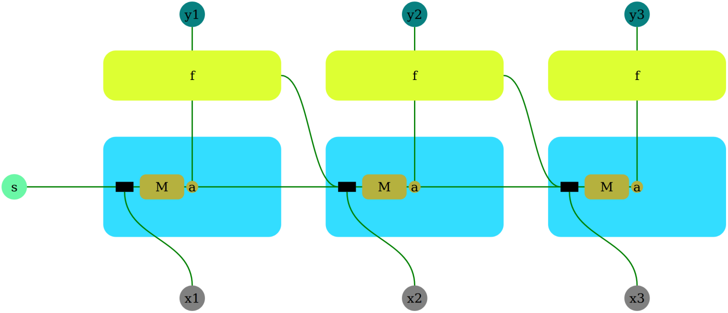


Figure: Each node is an operation. Black square represents concatenation, rest given from equation (1). a is an activiation function. The bias is not depicted in the graph, you may assume that it is part of the M operation. f is unspecified.

Vanilla RNN

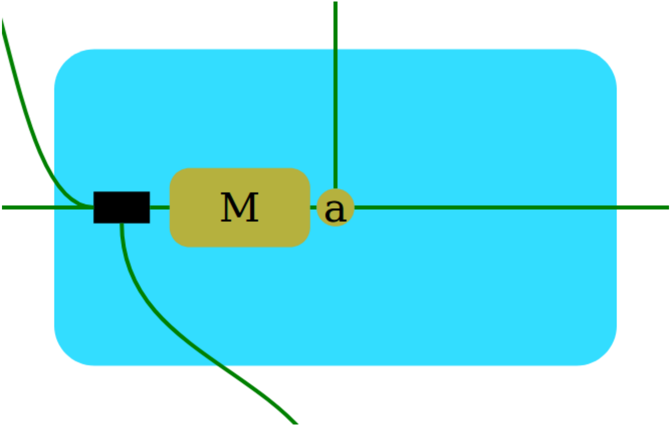


Figure: Each node is an operation. Black square represents concatenation, rest given from equation (1).

Preprocessing

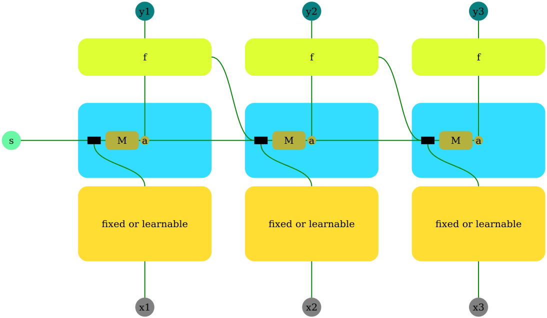


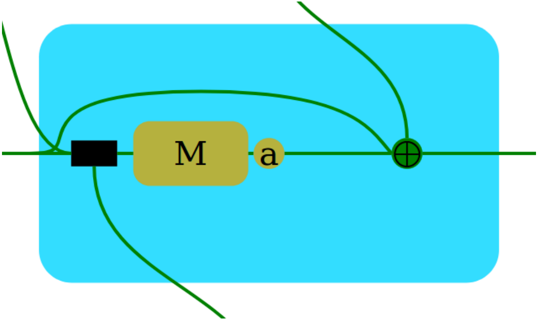
Figure: RNN preprocessing of input

- Both input and output can be preprocessed!

LSTM

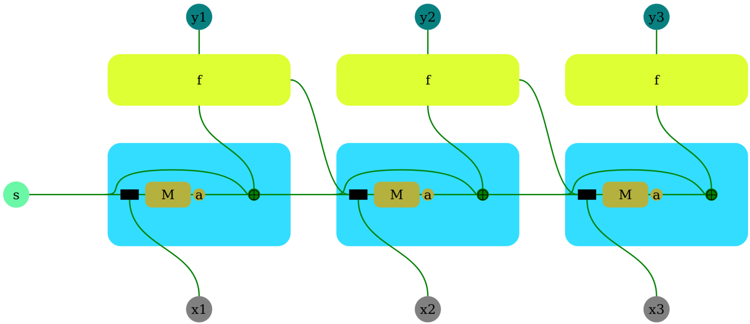
Residual / skip connection

Helps us store information.



$$r^t = a(U_r x^t + V_r s^{t-1} + W_r y^{t-1} + b_r)$$
$$s^t = s^{t-1} + r^t$$

Residual / skip connection



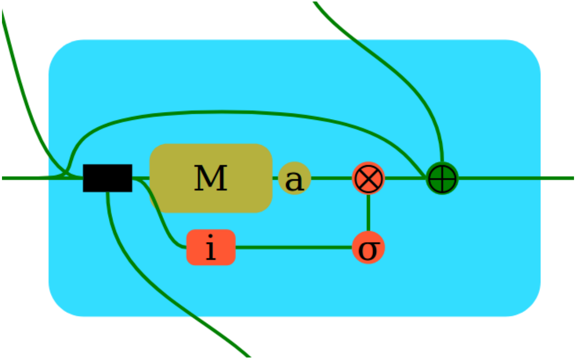
$$r^t = a(U_r x^t + V_r s^{t-1} + W_r y^{t-1} + b_r)$$
$$s^t = s^{t-1} + r^t$$

RNN w/residual in TensorFlow

```python
class RNNCell_v2(tf.keras.layers.AbstractRNNCell):
    def __init__(self, units):
        super(RNNCell_v2, self).__init__()
        self.units = units
        self.dense_r = layers.Dense(units)

    @property
    def state_size(self):
        return self.units

    def call(self, x, state):
        c = tf.concat([x, state], axis=-1)
        h = self.dense_r(c)
        # Should we add 'state' before or after activation?
        output = activations.tanh(h) + state # elementwise
        ↪  multiplication

        return output, output
```

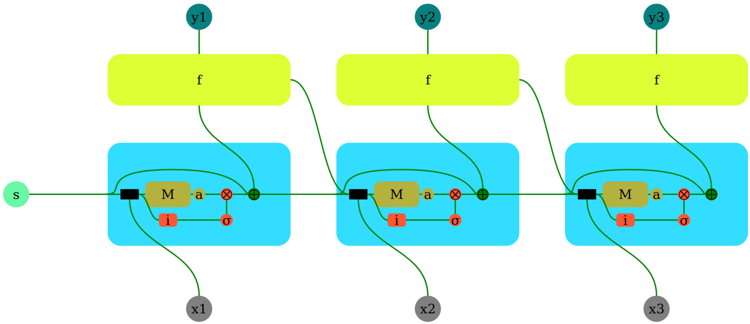
Input gate

Controls *write* access.



$$i^t = \sigma(U_i x^t + V_i s^{t-1} + W_i y^{t-1} + b_i)$$
$$s^t = s^{t-1} + i^t \odot r^t$$

Input gate



$$i^t = \sigma(U_i x^t + V_i s^{t-1} + W_i y^{t-1} + b_i)$$

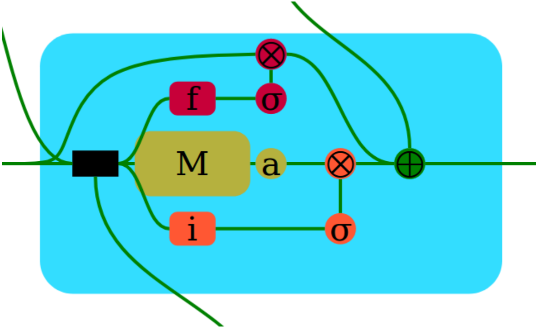
$$s^t = s^{t-1} + i^t \odot r^t$$

RNN w/input gate in TensorFlow

```
class RNNCell_v3(tf.keras.layers.AbstractRNNCell):
    def __init__(self, units):
        super(RNNCell_v3, self).__init__()
        self.units = units
        self.dense_r = layers.Dense(units)
        self.dense_i = layers.Dense(units)

    @property
    def state_size(self):
        return self.units

    def call(self, x, state):
        c = tf.concat([x, state], axis=-1)
        r = activations.tanh(self.dense_r(c))
        i = activations.sigmoid(self.dense_i(c))
        output = i*r + state # elementwise multiplication

        return output, output
```

Forget gate

Lets us forget things that are no longer useful.



$$f^t = \sigma(U_f x^t + V_f s^{t-1} + W_f y^{t-1} + b_f)$$
$$s^t = f^t \odot s^{t-1} + i^t \odot r^t$$

Forget gate

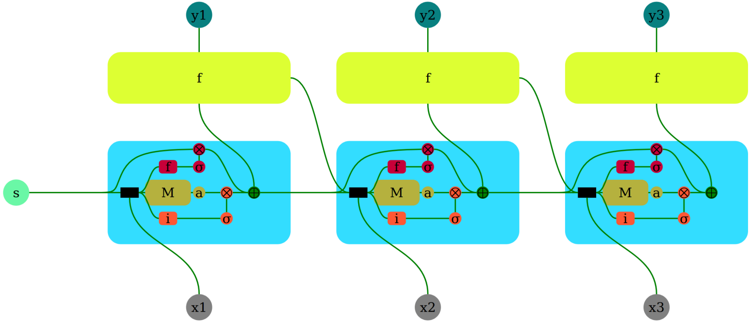


Figure: NOTE: The two f's are not related to each other!

$$f^t = \sigma(U_f x^t + V_f s^{t-1} + W_f y^{t-1} + b_f)$$
$$s^t = f^t \odot s^{t-1} + i^t \odot r^t$$

RNN w/forget gate in TensorFlow

```
class RNNCell_v4(tf.keras.layers.AbstractRNNCell):
    def __init__(self, units):
        super(RNNCell_v4, self).__init__()
        self.units = units
        self.dense_r = layers.Dense(units)
        self.dense_i = layers.Dense(units)
        self.dense_f = layers.Dense(units)

    @property
    def state_size(self):
      return self.units

    def call(self, x, state):
        c = tf.concat([x, state], axis=-1)
        r = activations.tanh(self.dense_r(c))
        i = activations.sigmoid(self.dense_i(c))
        f = activations.sigmoid(self.dense_f(c))
        output = i*r + f*state # elementwise multiplication
        return output, output
```

Output gate

Controls *read* access. Can be seen as an *attention* mechanism.

$$o^t = \sigma(U_o x^t + V_o s^{t-1} + W_o y^{t-1} + b_o)$$
$$\bar{s}^t = o^t \odot g(s^t)$$

- g is an activation function

LSTM in a slide

$$
\begin{aligned}
r^t &= a(U_r x^t + V_r \bar{s}^{t-1} + W_r y^{t-1} + b_r) \\
i^t &= \sigma(U_i x^t + V_i \bar{s}^{t-1} + W_i y^{t-1} + b_i) \\
f^t &= \sigma(U_f x^t + V_f \bar{s}^{t-1} + W_f y^{t-1} + b_f) \\
o^t &= \sigma(U_o x^t + V_o \bar{s}^{t-1} + W_o y^{t-1} + b_o) \\
s^t &= f^t \odot s^{t-1} + i^t \odot r^t \\
\bar{s}^t &= o^t \odot a(s^t) \\
y^t &= f(\bar{s}^t)
\end{aligned}
$$

LSTM in TensorFlow

```
class MyLSTMCell(tf.keras.layers.AbstractRNNCell):
    def __init__(self, units):
        super(MyLSTMCell, self).__init__()
        self.units = units
        self.dense_r = layers.Dense(units)
        self.dense_i = layers.Dense(units)
        self.dense_f = layers.Dense(units)
        self.dense_o = layers.Dense(units)

    def call(self, x, states):
        s, hidden_s = states
        c = tf.concat([x, s], axis=-1)
        r = activations.tanh(self.dense_r(c))
        i = activations.sigmoid(self.dense_i(c))
        f = activations.sigmoid(self.dense_f(c))
        o = activations.sigmoid(self.dense_o(c))
        hidden_s = i*r + f*hidden_s
        s = o*activations.tanh(hidden_s)
        return s, [s, hidden_s]
```

Differences to 'official' implementation

- Special initialization scheme used by default.
- Can choose different activation functions.
- Can choose different implementations!
- Options for dropout, weight constraints and regularization++

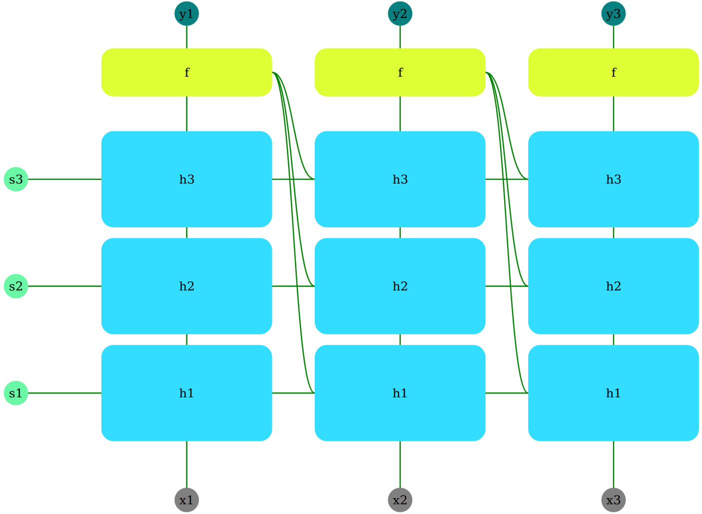
See `https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/keras/layers/recurrent.py` for more.

Depth in RNN

Multilayer perceptron

- Let h be a multilayer perceptron!
- If l layers, error propagation path will increase by factor l

Stacking RNNs



Complexity of RNN

What kind of complexity?

- Space: Memory usage
- Time: Number of serial steps
- Compute: FLOPs used

What kind of complexity?

- Space: Memory usage
- Time: Number of serial steps
- Compute: FLOPs used

Shall look at how these scales with sequence length

Complexity

Table: RNN complexity as a function sequence length

	Memory	Compute	Serial steps
Inference	O(1)	O(T)	O(T)
Training BPTT	O(T)	O(T)	O(T)
Training BPTT h(x, y*)	O(1)	O(T)	O(1)

Complexity

Table: RNN complexity as a function sequence length

	Memory	Compute	Serial steps
Inference	O(1)	O(T)	O(T)
Training BPTT	O(T)	O(T)	O(T)
Training BPTT h(x, y*)	O(1)	O(T)	O(1)

- Note that complexity for training depends on training algorithm!

A special case

- Only feed output to next time step (not state)
- During training we may use target values as input and thus parallelize training

$$s_t = h(x^t, y^{t-1})$$

Conclusion

Extensions:

- Next time!

Alternatives

- Convolutional neural networks
- Feedforward *attentional* networks