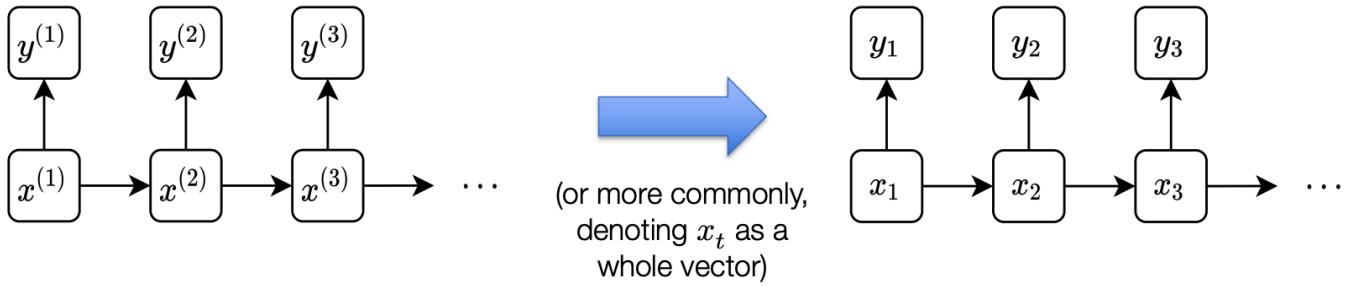
Sequence Modeling and Recurrent Networks

Sequence Modeling

在过去的预测模型中，输入输出对 $x^{(i)}, y^{(i)}$ 之间相互独立 (independent identically distributed, i.i.d.)，不存在顺序关系。

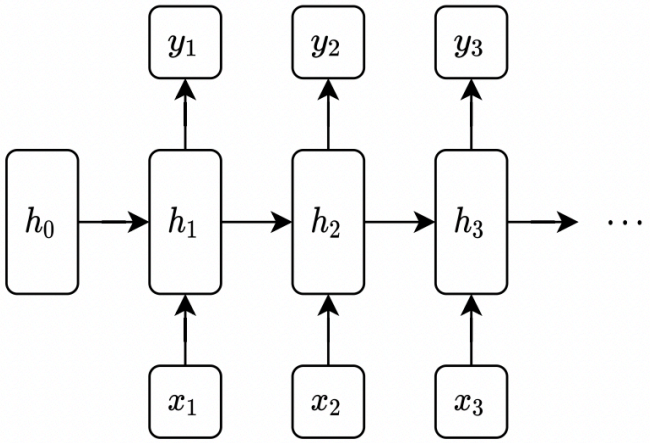
但在输入输出对为 sequence 的时候，需要用到顺序关系的信息来进行预测，即当前输出取决于当前输入以及过往输入/输出。



典型应用：Speech tagging, Speech to text, Translating, Autoregressive prediction

Recurrent Neural Networks

RNN 在输入与输出之间维护了 hidden layer, 记录了过往输入的信息。输出由前一个 hidden layer 和当前输入共同产生。



$$h_t = f(W_{hh}h_{t-1} + W_{hx}x_t + b_h)$$
$$y_t = g(W_{yh}h_t + b_y)$$

where f and g are activation functions, W_{hh}, W_{hx}, W_{yh} are weights and b_h, b_y are bias terms

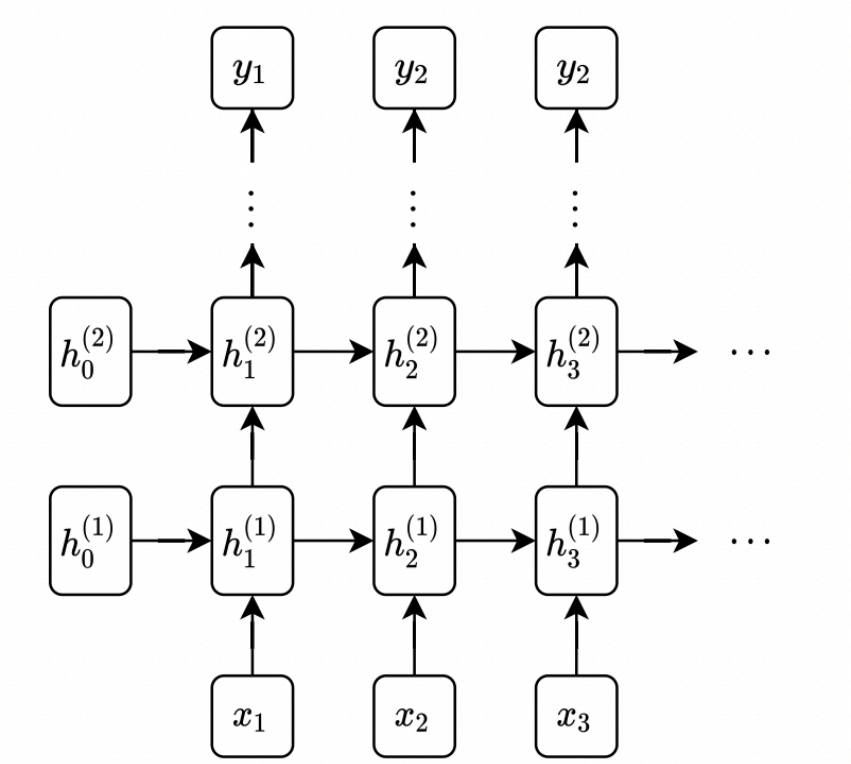
其中 f 和 g 是非线性的激活函数。

给定输入和目标输出 $(x_1, \ldots, x_T, y_1^*, \ldots, y_T^*)$, RNN 训练过程如下，主要依赖 AD 来更新参数：

```
opt = Optimizer(params = (W_hh, W_hx, W_yh, b_h, b_y))
h[0] = 0
l = 0
for t = 1,...,T:
   h[t] = f(W_hh * h[t-1] + W_hx * x[t] + b_h)
   y[t] = g(W_yh * h[t] + b_y)
   l += Loss(y[t], y_star[t])
l.backward()
opt.step()
```

Stacking RNNs

与传统神经网络类似，RNN 也可以进行堆叠加深层数，但与其他架构相比，RNN 的深度并没那么重要。

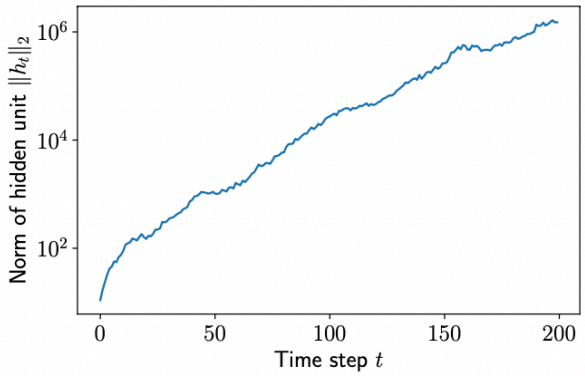


Problems of RNN

RNN 训练的问题与之前 MLP 的问题类似，即激活函数值和梯度随深度增加而放大/缩减的问题。

对于训练 long sequence 的 RNN, 如果权重参数初始化不恰当，会导致 hidden activations 和梯度爆炸/消减。

Exploding Activation:

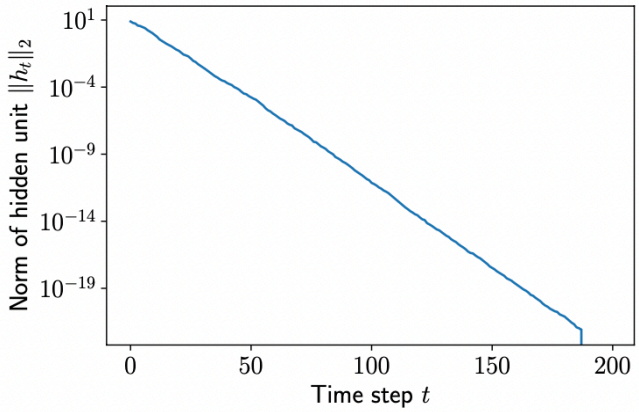


Single layer RNN with ReLU activations, using weight initialization

$$W_{hh} \sim \mathcal{N}(0, 3/n)$$

Recall that $\sigma^2 = 2/n$ was the "proper" initialization for ReLU activations

Vanishing Activation:



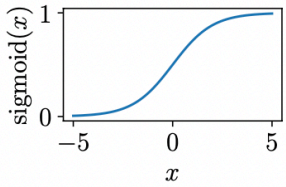
Single layer RNN with ReLU activations, using weight initialization

$$W_{hh} \sim \mathcal{N}(0, 1.5/n)$$

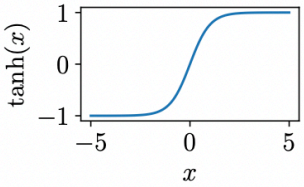
Non-zero input only provided here for time 1, showing decay of information about this input over time

Alternative Activations

ReLU 的一个问题在于它的增长是不受限的，因此可尝试替换激活函数为 sigmoid 或 tanh:



$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$$



$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

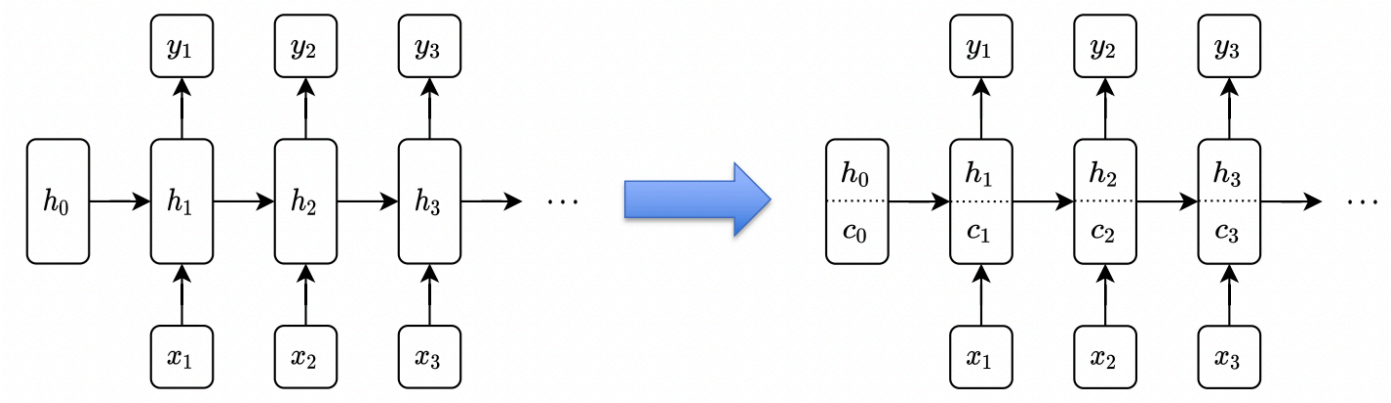
但是纯粹替换激活函数仍然无法解决消减的问题，以 tanh 为例，在接近饱和的位置 (tanh(x) = 1 or -1), 梯度接近 0, 在梯度最大的位置，函数值为 0.

LSTMs

LSTM: Long Short Term Memory

LSTM 能解决上述中传统 RNN 的问题

LSTM 将 hidden unit 划分为两个部分，即 hidden state 和 cell state. 二者同属于 hidden unit.



h_t 和 c_t 的算式如下：

$$\begin{bmatrix} i_t \\ f_t \\ g_t \\ o_t \end{bmatrix} = \begin{pmatrix} \text{sigmoid} \\ \text{sigmoid} \\ \tanh \\ \text{sigmoid} \end{pmatrix} (W_{hh} h_{t-1} + \mathrm{W}_{hx} x_t + b_h)$$

$$c_t = c_{t-1} \circ f_t + i_t \circ g_t$$

$$h_t = \tanh(c_t) \circ o_t$$

其中 i_t 称为 input gate, f_t 称为 forget gate, o_t 称为 output gate.

对 LSTM 的理解：$c_t = c_{t-1} \circ f_t + i_t \circ g_t$ 中的 c_{t-1} 代表前一个 hidden unit 的结果，而 $f_t \in [0,1]^n$, 实际上是控制过往结果的影响，类似记忆的衰减。 形象地说，c_t 包含了过去的记忆以及现在输入的新记忆。

对 f_t 而言，在 sigmoid 函数的饱和点意味着不改变 c_{t-1}. 因此，LSTM 不会导致梯度消减的问题。

Beyond Simple Sequential Models

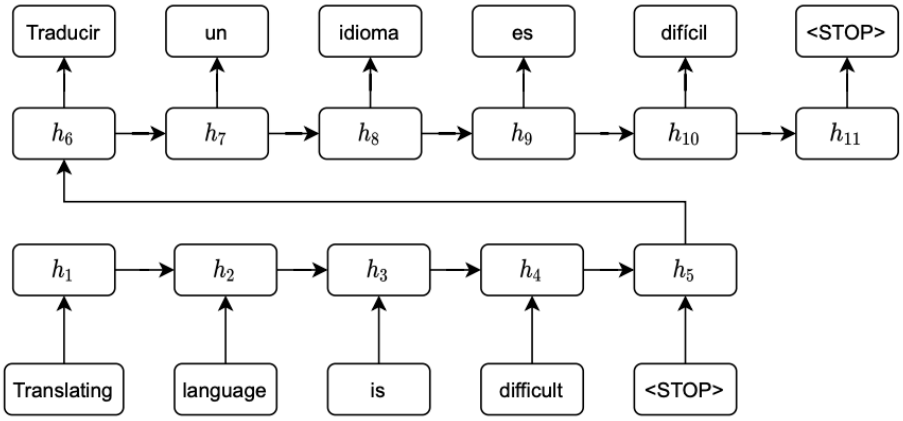
很多实际场景中，对于当前输入，不仅要考虑先前输入，还希望考虑未来的输入。

Seq2seq Model

在语言翻译中，通常希望在翻译时考虑整个句子，因此可以将两个 RNN 结合起来，一个作为 encoder, 另一个作为 decoder.

Encoder 负责处理 sequence 以生成最后的 hidden state, 没有 loss function

Decoder 将 encoder 最后的 hidden state 作为输入，该状态包含了整个 sequence 的信息，生成所需的输出。



Bidirectional RNNs

核心思想：将一个正向运行的 RNN 与一个反向运行的 RNN 堆叠，使整个 sequence 的信息被传播到 hidden state 中。