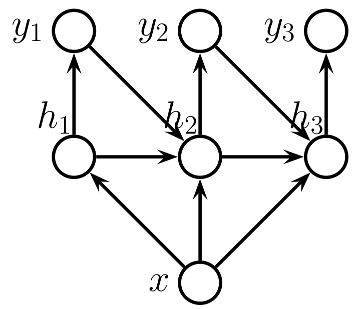

Figure 15.1: Recurrent neural network (RNN) for generating a variable length output sequence $\boldsymbol{y}_{1:T}$ given an optional fixed length input vector $\boldsymbol{x}$.

15.2.1.1 Models

For notational simplicity, let T be the length of the output (with the understanding that this is chosen dynamically). The RNN then corresponds to the following conditional generative model:

$$p(\boldsymbol{y}_{1:T}|\boldsymbol{x}) = \sum_{\boldsymbol{h}_{1:T}} p(\boldsymbol{y}_{1:T}, \boldsymbol{h}_{1:T}|\boldsymbol{x}) = \sum_{\boldsymbol{h}_{1:T}} \prod_{t=1}^{T} p(\boldsymbol{y}_t|\boldsymbol{h}_t) p(\boldsymbol{h}_t|\boldsymbol{h}_{t-1}, \boldsymbol{y}_{t-1}, \boldsymbol{x}) \tag{15.1}$$

where $\boldsymbol{h}_t$ is the hidden state, and where we define $p(\boldsymbol{h}_1|\boldsymbol{h}_0, \boldsymbol{y}_0, \boldsymbol{x}) = p(\boldsymbol{h}_1|\boldsymbol{x})$ as the initial hidden state distribution (often deterministic).

The output distribution is usually given by

$$p(\boldsymbol{y}_t|\boldsymbol{h}_t) = \text{Cat}(\boldsymbol{y}_t|\text{softmax}(\mathbf{W}_{hy}\boldsymbol{h}_t + \boldsymbol{b}_y)) \tag{15.2}$$

where $\mathbf{W}_{hy}$ are the hidden-to-output weights, and $\boldsymbol{b}_y$ is the bias term. However, for real-valued outputs, we can use

$$p(\boldsymbol{y}_t|\boldsymbol{h}_t) = \mathcal{N}(\boldsymbol{y}_t|\mathbf{W}_{hy}\boldsymbol{h}_t + \boldsymbol{b}_y, \sigma^2\mathbf{I}) \tag{15.3}$$

We assume the hidden state is computed deterministically as follows:

$$p(\boldsymbol{h}_t|\boldsymbol{h}_{t-1}, \boldsymbol{y}_{t-1}, \boldsymbol{x}) = \mathbb{I}\left(\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{y}_{t-1}, \boldsymbol{x})\right) \tag{15.4}$$

for some deterministic function f. The update function f is usually given by

$$\boldsymbol{h}_t = \varphi(\mathbf{W}_{xh}[\boldsymbol{x}; \boldsymbol{y}_{t-1}] + \mathbf{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h) \tag{15.5}$$

where $\mathbf{W}_{hh}$ are the hidden-to-hidden weights, $\mathbf{W}_{xh}$ are the input-to-hidden weights, and $\boldsymbol{b}_h$ are the bias terms. See Figure 15.1 for an illustration, and rnn_jax.ipynb for some code.

Note that $\boldsymbol{y}_t$ depends on $\boldsymbol{h}_t$, which depends on $\boldsymbol{y}_{t-1}$, which depends on $\boldsymbol{h}_{t-1}$, and so on. Thus $\boldsymbol{y}_t$ implicitly depends on all past observations (as well as the optional fixed input $\boldsymbol{x}$). Thus an RNN overcomes the limitations of standard Markov models, in that they can have unbounded memory. This makes RNNs theoretically as powerful as a **Turing machine** [SS95; PMB19]. In practice,