

- Recurrent networks that produce an output at each time step and have recurrent connections between hidden units, illustrated in figure 10.3.
- Recurrent networks that produce an output at each time step and have recurrent connections only from the output at one time step to the hidden units at the next time step, illustrated in figure 10.4
- Recurrent networks with recurrent connections between hidden units, that read an entire sequence and then produce a single output, illustrated in figure 10.5.

figure 10.3 is a reasonably representative example that we return to throughout most of the chapter.

The recurrent neural network of figure 10.3 and equation 10.8 is universal in the sense that any function computable by a Turing machine can be computed by such a recurrent network of a finite size. The output can be read from the RNN after a number of time steps that is asymptotically linear in the number of time steps used by the Turing machine and asymptotically linear in the length of the input (Siegelmann and Sontag, 1991; Siegelmann, 1995; Siegelmann and Sontag, 1995; Hyotyniemi, 1996). The functions computable by a Turing machine are discrete, so these results regard exact implementation of the function, not approximations. The RNN, when used as a Turing machine, takes a binary sequence as input and its outputs must be discretized to provide a binary output. It is possible to compute all functions in this setting using a single specific RNN of finite size (Siegelmann and Sontag (1995) use 886 units). The “input” of the Turing machine is a specification of the function to be computed, so the same network that simulates this Turing machine is sufficient for all problems. The theoretical RNN used for the proof can simulate an unbounded stack by representing its activations and weights with rational numbers of unbounded precision.

We now develop the forward propagation equations for the RNN depicted in figure 10.3. The figure does not specify the choice of activation function for the hidden units. Here we assume the hyperbolic tangent activation function. Also, the figure does not specify exactly what form the output and loss function take. Here we assume that the output is discrete, as if the RNN is used to predict words or characters. A natural way to represent discrete variables is to regard the output \mathbf{o} as giving the unnormalized log probabilities of each possible value of the discrete variable. We can then apply the softmax operation as a post-processing step to obtain a vector $\hat{\mathbf{y}}$ of normalized probabilities over the output. Forward propagation begins with a specification of the initial state $\mathbf{h}^{(0)}$. Then, for each time step from