Recurrent Neurons

Up to now we have mostly looked at feedforward neural networks, where the activations flow only in one direction, from the input layer to the output layer (except for a few networks in Appendix E). A recurrent neural network looks very much like a feedforward neural network, except it also has connections pointing backward. Let's look at the simplest possible RNN, composed of just one neuron receiving inputs, producing an output, and sending that output back to itself, as shown in Figure 14-1 (left). At each *time step t* (also called a *frame*), this *recurrent neuron* receives the inputs $\mathbf{x}_{(t)}$ as well as its own output from the previous time step, $y_{(t-1)}$. We can represent this tiny network against the time axis, as shown in Figure 14-1 (right). This is called *unrolling the network through time*.

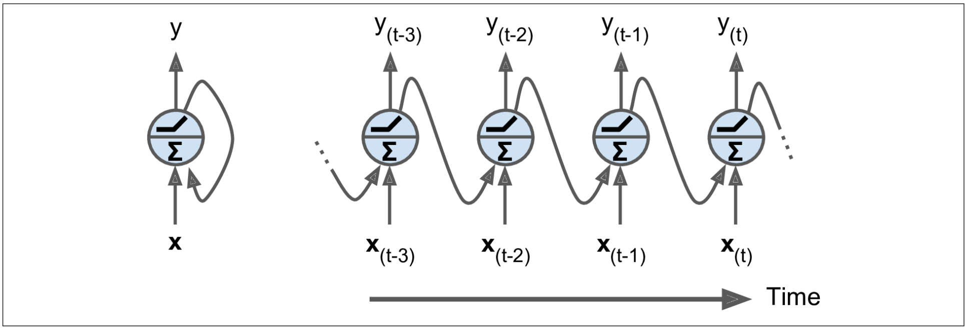

Figure 14-1. A recurrent neuron (left), unrolled through time (right)

You can easily create a layer of recurrent neurons. At each time step *t*, every neuron receives both the input vector $\mathbf{x}_{(t)}$ and the output vector from the previous time step $\mathbf{y}_{(t-1)}$, as shown in Figure 14-2. Note that both the inputs and outputs are vectors now (when there was just a single neuron, the output was a scalar).

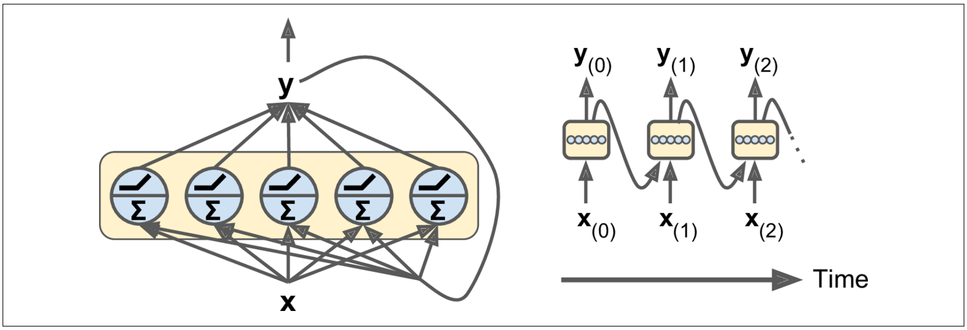

Figure 14-2. A layer of recurrent neurons (left), unrolled through time (right)

Each recurrent neuron has two sets of weights: one for the inputs $\mathbf{x}_{(t)}$ and the other for the outputs of the previous time step, $\mathbf{y}_{(t-1)}$. Let's call these weight vectors $\mathbf{w}_x$ and $\mathbf{w}_y$.

The output of a single recurrent neuron can be computed pretty much as you might expect, as shown in Equation 14-1 (b is the bias term and $\phi(\cdot)$ is the activation function, e.g., ReLU[1]).

Equation 14-1. Output of a single recurrent neuron for a single instance

$$\mathbf{y}_{(t)} = \phi\left(\mathbf{x}_{(t)}{}^{T} \cdot \mathbf{w}_{x} + \mathbf{y}_{(t-1)}{}^{T} \cdot \mathbf{w}_{y} + b\right)$$

Just like for feedforward neural networks, we can compute a whole layer's output in one shot for a whole mini-batch using a vectorized form of the previous equation (see Equation 14-2).

Equation 14-2. Outputs of a layer of recurrent neurons for all instances in a mini-batch

$$\begin{aligned}\mathbf{Y}_{(t)} &= \phi\left(\mathbf{X}_{(t)} \cdot \mathbf{W}_{x} + \mathbf{Y}_{(t-1)} \cdot \mathbf{W}_{y} + \mathbf{b}\right)\\ &= \phi\left(\begin{bmatrix}\mathbf{X}_{(t)} & \mathbf{Y}_{(t-1)}\end{bmatrix} \cdot \mathbf{W} + \mathbf{b}\right) \text{ with } \mathbf{W} = \begin{bmatrix}\mathbf{W}_{x}\\ \mathbf{W}_{y}\end{bmatrix}\end{aligned}$$

- $\mathbf{Y}_{(t)}$ is an $m \times n_{\text{neurons}}$ matrix containing the layer's outputs at time step t for each instance in the mini-batch (m is the number of instances in the mini-batch and n_{neurons} is the number of neurons).
- $\mathbf{X}_{(t)}$ is an $m \times n_{\text{inputs}}$ matrix containing the inputs for all instances (n_{inputs} is the number of input features).
- $\mathbf{W}_{x}$ is an $n_{\text{inputs}} \times n_{\text{neurons}}$ matrix containing the connection weights for the inputs of the current time step.
- $\mathbf{W}_{y}$ is an $n_{\text{neurons}} \times n_{\text{neurons}}$ matrix containing the connection weights for the outputs of the previous time step.
- The weight matrices $\mathbf{W}_{x}$ and $\mathbf{W}_{y}$ are often concatenated into a single weight matrix $\mathbf{W}$ of shape $(n_{\text{inputs}} + n_{\text{neurons}}) \times n_{\text{neurons}}$ (see the second line of Equation 14-2).
- $\mathbf{b}$ is a vector of size n_{neurons} containing each neuron's bias term.

[1] Note that many researchers prefer to use the hyperbolic tangent (tanh) activation function in RNNs rather than the ReLU activation function. For example, take a look at by Vu Pham et al.'s paper "Dropout Improves Recurrent Neural Networks for Handwriting Recognition". However, ReLU-based RNNs are also possible, as shown in Quoc V. Le et al.'s paper "A Simple Way to Initialize Recurrent Networks of Rectified Linear Units".