The rest of the code is the same as earlier. This can provide a significant speed boost since there is just one fully connected layer instead of one per time step.

Creative RNN

Now that we have a model that can predict the future, we can use it to generate some creative sequences, as explained at the beginning of the chapter. All we need is to provide it a seed sequence containing `n_steps` values (e.g., full of zeros), use the model to predict the next value, append this predicted value to the sequence, feed the last `n_steps` values to the model to predict the next value, and so on. This process generates a new sequence that has some resemblance to the original time series (see Figure 14-11).

```
sequence = [0.] * n_steps
for iteration in range(300):
    X_batch = np.array(sequence[-n_steps:]).reshape(1, n_steps, 1)
    y_pred = sess.run(outputs, feed_dict={X: X_batch})
    sequence.append(y_pred[0, -1, 0])
```

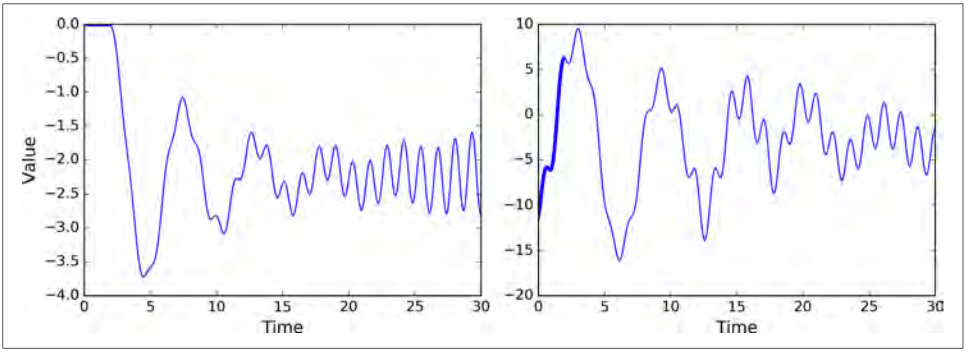

Figure 14-11. Creative sequences, seeded with zeros (left) or with an instance (right)

Now you can try to feed all your John Lennon albums to an RNN and see if it can generate the next "Imagine." However, you will probably need a much more powerful RNN, with more neurons, and also much deeper. Let's look at deep RNNs now.

Deep RNNs

It is quite common to stack multiple layers of cells, as shown in Figure 14-12. This gives you a *deep RNN*.