Training the Recurrent Network: Backpropagation Through Time

The recurrent neural network is trained in much the same way as other traditional neural networks by using the backpropagation algorithm. However, the backpropagation algorithm is modified into what is called backpropagation through time (BPTT).

Due to the architectural loop or recurrent structure of the recurrent network, vanilla backpropagation as is cannot work. Training a network using backpropagation involves calculating the error gradient, moving backward from the output layer through the hidden layers of the network and adjusting the network weights. However, this operation cannot work in the recurrent neuron because we have just one neural cell with recurrent connections to itself.

So, in order to train the recurrent network using backpropagation, we unroll the recurrent neuron across the time instants and apply backpropagation to the unrolled neurons at each time layer the same way it is done for a traditional feedforward neural network. This operation is further illustrated in Figure 36-12.

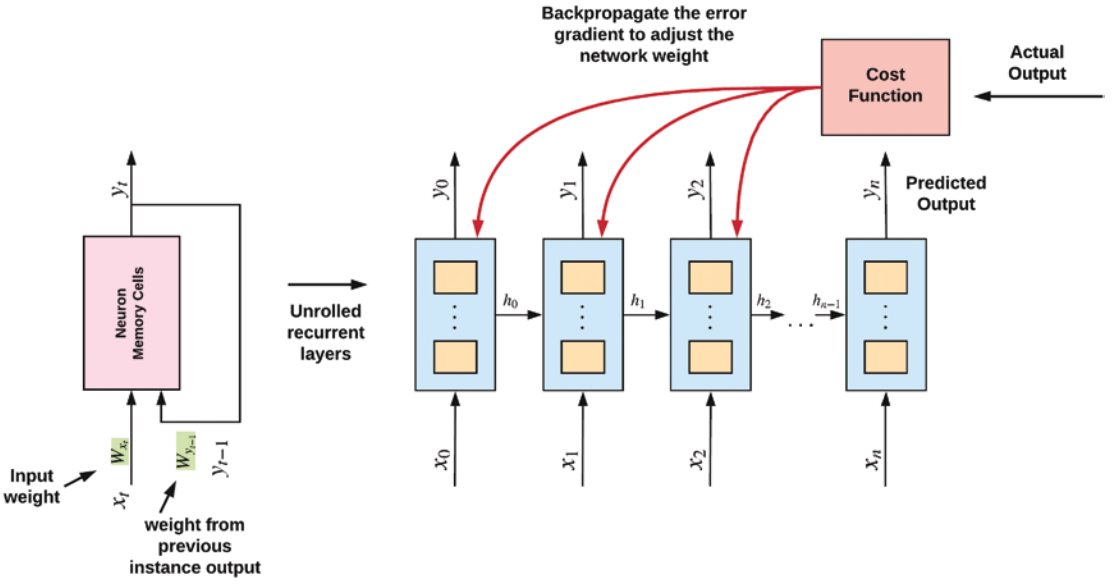

Figure 36-12. *Backpropagation through time*

A significant challenge of training the recurrent neural network is the vanishing and exploding gradient problem. When training a deep recurrent network for many layers of time instants, calculating the gradients of the weights of the neurons can become very volatile. When this happens, the value of the gradient can become extremely large tending to infinity, or they become tiny, all the way to zero. When this happens, the neurons become dead and cannot train or learn any new information further. This effect is called the exploding and vanishing gradient problem.

The exploding and vanishing gradient problem is most prevalent in recurrent neural networks because of the long-term dependencies or time instant of the unrolled recurrent neuron. A proposed alternative technique for mitigating this problem in recurrent networks (in addition to other discussed methods such as gradient clipping, batch normalization, and using a non-saturating activation function such as ReLu) is to discard early time instances or time instances in the distant past. This technique is called Truncated Backpropagation Through Time (truncated BPTT).

However, truncated BPTT suffers a significant drawback, and this is that some problems rely heavily on long-term dependencies to be able to make a prediction. A typical example is in language modeling where the long-term sequence of words in the past is vital in predicting the next word in the sequence.

The shortcoming of truncated BPTT and the need to deal with the problem of exploding and vanishing gradients led to the development of a memory cell called the Long Short-Term Memory or LSTM for short, which can store the long-term information of the problem in the memory cell of the recurrent network.

The Long Short-Term Memory (LSTM) Network

Long Short-Term Memory (LSTM) belongs to a class of RNN called gated recurrent unit. They are called *gated* because unlike the basic recurrent units, they contain extra components called gates that control the flow of information within the recurrent cell. This includes choosing what information to store in the cell and what information to discard or forget.

LSTM is very efficient for capturing the long-term dependencies across a large number of time instants. It does this by having a slightly more sophisticated cell than the basic recurrent units. The components of the LSTM are the

- Memory cell
- Input gate