1 Introduction

This is a quick and dirty introduction to recurrent neural networks (RNNs) written mainly for my own sake.

RNNs are functions applied to sequences that output other sequences; typically (always?) the output is the predicted continuation of the input sequence. RNNs have been used for natural language processing and music generation.

RNNs consist of a NN applied to each element in the input sequence where the NN also has inter-element connections. The same network is applied to each element, hence the name recurrent. This is not essential but at least drastically reduces the number of parameters to learn; it also solves the hard problem of how to define (not to mention train!) a potentially infinite set of NNs. A schematic of a RNN is show in the figure.

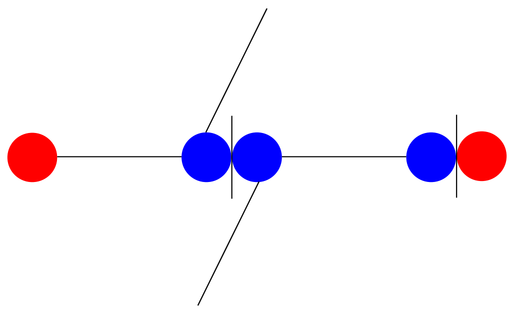

Figure 1: The basis neural net unit of a RNN.

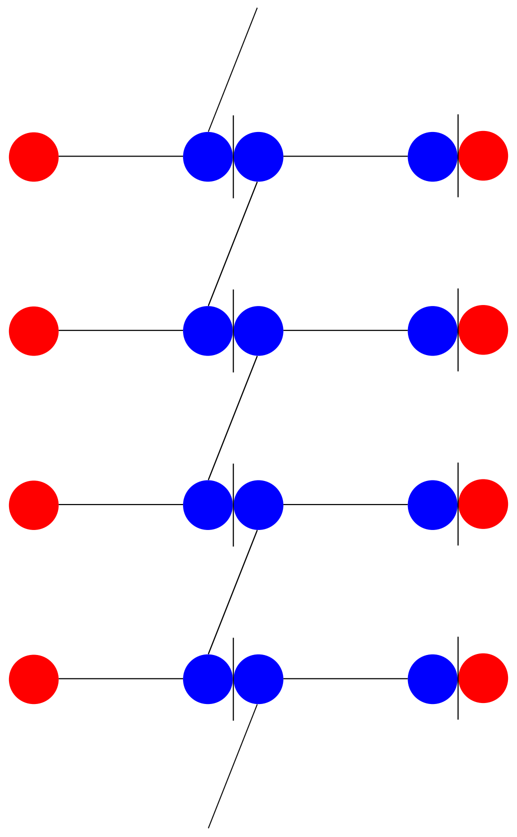

Figure 2: The RNN applied to a sequence of length 4.

Each step, numbered by t, in the sequence has an input variable, x_t (indicated by the red circles on the left side), a hidden state pre-activation a_t (blue cicles on the left of the vertical line), a hidden state post-activation s_t (blue circles on the right of the vertical line), an output state pre-activation, p_t (blue circle on the right) and an output state post-activation o_t (red circle on the right). For our purposes the output o_t is interpreted as probabilities that the RNN assigns to the next element of the sequence, i.e. at step $t + 1$. The diagrams above are intended to illustrate that these numbers are calculated as

$$a_t = Ux_t + Ws_{t-1}, \tag{1.1}$$

$$s_t = \mathcal{A}(a_t), \tag{1.2}$$

$$p_t = Vs_t, \tag{1.3}$$

$$o_t = \mathcal{O}(p_t). \tag{1.4}$$

Here $\mathcal{A}, \mathcal{O}$ are the activation and output activation functions, e.g. $\tanh(x)$ and "softmax" for word processing RNNs. For the first element of the sequence one must choose an initialization of s_{-1}, e.g. $s_{-1} = 0$. We may only be interested in the output of the last element of the RNN, e.g. when we try to predict or generate the next word in a sentence or the next musical note.

2 Training of a RNN

As with regular NNs training relies on formulas for the gradients calculated using the chain rule. In the case of RNNs the expressions are slightly more complicated because the same matrix appears in several places. E.g. a_t depends on U both through Ux_t and Ws_{t-1}. We assume that the error can be written as

$$E = \sum_t E_t, \tag{2.1}$$

which is not very restrictive.

Let us calculate the gradients $\frac{\partial E_t}{\partial V}, \frac{\partial E_t}{\partial W}, \frac{\partial E_t}{\partial U}$. Since E_t depends only on the output of the RNN (and the target output but that is obviously independent of the parameters of the network) we can write

$$\frac{\partial E_t}{\partial V} = \frac{\partial E_t}{\partial p_t}\frac{\partial p_t}{\partial V}. \tag{2.2}$$

Note that p_t is a vector and V is a matrix; if we write the above equation with indices we have

$$\frac{\partial E_t}{\partial V_{ij}} = \sum_k \frac{\partial E_t}{\partial (p_t)_k}\frac{\partial (p_t)_k}{\partial V_{ij}}. \tag{2.3}$$

But let us just view $\frac{\partial E_t}{\partial p_t}$ as a covariant/column vector and $\frac{\partial p_t}{\partial V_{ij}}$ as a contravariant/column vector. Now, assuming we use the canonical link we have

$$\frac{\partial E_t}{\partial p_t} = o_t - t_t, \tag{2.4}$$

where t_t is the target vector. Finally (for this gradient)

$$\frac{\partial p_t}{\partial V} = \frac{\partial V s_t}{\partial V} = s_t + V\frac{\partial s_t}{\partial V} = s_t + 0 = s_t. \tag{2.5}$$

I have written every step explicitly to emphasize that the expression is only simple because s_t does not depend on V:

$$\frac{\partial E_t}{\partial V} = (o_t - t_t) \otimes s_t^T. \tag{2.6}$$

Here I have also emphasized that we end up with a matrix:

$$\frac{\partial}{\partial A} x^T A y = \frac{\partial}{\partial A} \mathrm{Tr} A y x^T = x y^T. \tag{2.7}$$

Next up:

$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial p_t}\frac{\partial p_t}{\partial W} = \frac{\partial E_t}{\partial p_t}\frac{\partial p_t}{\partial s_t}\frac{\partial s_t}{\partial W} \tag{2.8}$$

$$= \frac{\partial E_t}{\partial p_t}\frac{\partial p_t}{\partial s_t}\frac{\partial s_t}{\partial a_t}\frac{\partial a_t}{\partial W} = \frac{\partial E_t}{\partial p_t}\frac{\partial p_t}{\partial s_t}\frac{\partial s_t}{\partial a_t}\frac{\partial (Ux_t + Ws_{t-1})}{\partial W}$$

$$= \frac{\partial E_t}{\partial p_t}\frac{\partial p_t}{\partial s_t}\frac{\partial s_t}{\partial a_t}\left(s_{t-1} + W\frac{\partial s_{t-1}}{\partial W}\right). \tag{2.9}$$

The form of the equation is now similar to the first line eq. (2.8) (which is why it got its own label). To calculate the last factor we use the structure we already uncovered:

$$\frac{\partial s_t}{\partial W} = \frac{\partial s_t}{\partial a_t}\left(s_{t-1} + W\frac{\partial s_{t-1}}{\partial W}\right). \tag{2.10}$$

Applying this formula until we reach the first layer $t = 0$ allows us to calculate the full gradient $\partial_W E_t$. Just so we all agree on the matrix structure we are working with here, let's state it explicitly: $\frac{\partial s_t}{\partial a_t}$ is a matrix with entries

$$\left(\frac{\partial s_t}{\partial a_t}\right)_{ij} = \frac{\partial (s_t)_i}{\partial (a_t)_j}, \tag{2.11}$$

s_{t-1}, $\frac{\partial s_{t-1}}{\partial W}$ are vectors.

Next in line:

$$\frac{\partial E_t}{\partial U} = \frac{\partial E_t}{\partial p_t}\frac{\partial p_t}{\partial s_t}\frac{\partial s_t}{\partial a_t}\frac{\partial (Ux_t + Ws_{t-1})}{\partial U} = \frac{\partial E_t}{\partial p_t}\frac{\partial p_t}{\partial s_t}\frac{\partial s_t}{\partial a_t}\left(x_t + W\frac{\partial s_{t-1}}{\partial U}\right) \tag{2.12}$$

Here the key formula is

$$\frac{\partial s_t}{\partial U} = \frac{\partial s_t}{\partial a_t}\left(x_t + W\frac{\partial s_{t-1}}{\partial U}\right). \tag{2.13}$$

3 LSTMs

There are certain problems with regular RNNs. Gradients tend to go to zero as we move back along the sequence. This means that during training the output at T is not very sensitive to the input at $t \ll T$. Obviously this is very bad for text generators or for musical nets that want to do more than just meander through a scale. I won't go into detail on the "vanishing gradient" problem, but take a look at wildml.com if you are interested.

Long-Short-term memory [neural networks] (LSTMs) combat this problem. LSTMs calculate the hidden state in a different way that intuitively allows information to more easily propagate from step to step. We introduce a "memory" at each step which is influenced by the input at the current step, the memory at the previous step as well as the hidden state at the previous step. Concretely the hidden state, s_t is now calculate by

$$i_t = \sigma(U^i x_t + W^i s_{t-1}), \tag{3.1}$$
$$f_t = \sigma(U^f x_t + W^f s_{t-1}), \tag{3.2}$$
$$o_t = \sigma(U^o x_t + W^o s_{t-1}), \tag{3.3}$$
$$g_t = \mathcal{A}(U^g x_t + W^g s_{t-1}), \tag{3.4}$$
$$c_t = c_{t-1} \circ f_t + g_t \circ i_t, \tag{3.5}$$
$$s_t = \mathcal{A}(c_t) \circ o_t, \tag{3.6}$$
$$y_t = \mathcal{O}(V s_t). \tag{3.7}$$

We have used $\circ$ to indicate element-wise multiplication, $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function. We have also redefined what o means; this is just to match the notation on wildml.com. The output is now denoted by y_t. It is quite complicated and can be illustrated like in the figure below.

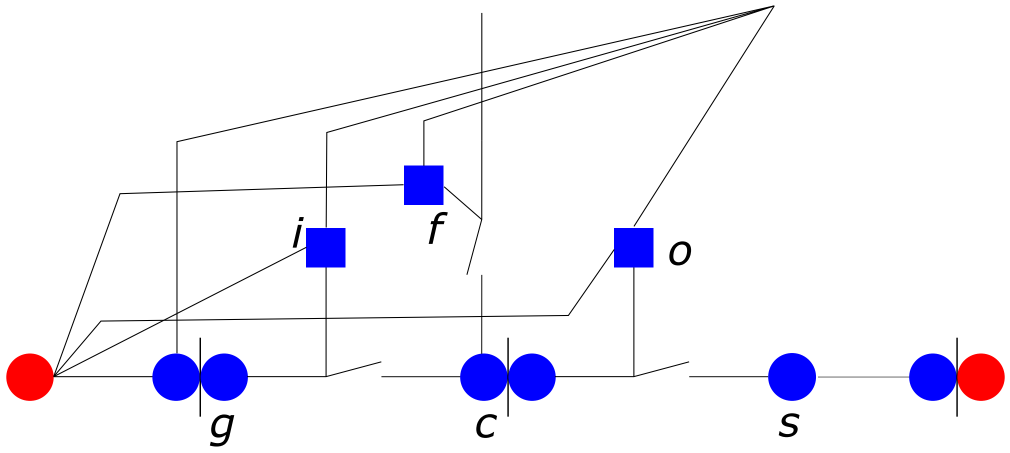

Figure 3

The derivate with respect to V is easy so let's skip that.
To calculate the other gradients let me introduce some new variables.

$$d_t = \mathcal{A}(c_t), \tag{3.8}$$
$$a_t = U^g x_t + W^g s_{t-1}, \tag{3.9}$$
$$u_t = U^o x_t + W^o s_{t-1} \tag{3.10}$$

$$\begin{aligned}
\frac{\partial E_t}{\partial W^o} &= \frac{\partial E_t}{\partial p_t}\frac{\partial p_t}{\partial W^o} = \frac{\partial E_t}{\partial p_t}\frac{\partial p_t}{\partial s_t}\frac{\partial s_t}{\partial W^o} \\
&= \frac{\partial E_t}{\partial p_t}\frac{\partial p_t}{\partial s_t}\frac{\partial (d_t \circ o_t)}{\partial W^o} \\
&= \frac{\partial E_t}{\partial p_t}\frac{\partial p_t}{\partial s_t}\left(\frac{\partial d_t}{\partial W^o} \circ o_t + d_t \circ \frac{\partial o_t}{\partial W^o}\right) \\
&= \frac{\partial E_t}{\partial p_t}\frac{\partial p_t}{\partial s_t}\left(\frac{\partial d_t}{\partial W^o} \circ o_t + d_t \circ \frac{\partial o_t}{\partial W^o}\right) \\
&= \frac{\partial E_t}{\partial p_t}\frac{\partial p_t}{\partial s_t}\left(\frac{\partial d_t}{\partial W^o} \circ o_t + d_t \circ \frac{\partial o_t}{\partial u_t}\frac{\partial u_t}{\partial s_{t-1}}\frac{\partial s_{t-1}}{\partial W^o}\right).
\end{aligned} \tag{3.11}$$