**Report**

This report explains the basic logic behind each major step of RNN implementation and shows the training results of sentiment analsyis using RNN.

**1. RNN Step Forward**

Given input data $x_t$ at timestep t, hidden state from previous timestep prev_h ($h_{t-1}$), $W_x$ , $W_h$ and b, we calculate next_h ($h_t$) by

$$\mathrm{a}_t = W_x\, x_t + W_h\, h_{t-1} + b$$
$$h_t = \tanh(\mathrm{a}_t)$$

**2. RNN Step Backward**

Given dnext_h ($\mathrm{dh}_t$) , gradient of loss with respect to next hidden state t, and with $\frac{\mathrm{dh}_t}{\mathrm{d}a_t} = 1 - {h_t}^2$, we have

$$\mathrm{d}x_t = \mathrm{dh}_t \frac{\partial \mathrm{h}_t}{\partial a_t}\frac{\partial \mathrm{a}_t}{\partial x_t} = W_x^T \mathrm{dh}_t\,(1 - {h_t}^2)$$
$$\mathrm{d}h_{t-1} = \mathrm{dh}_t \frac{\partial \mathrm{h}_t}{\partial a_t}\frac{\partial \mathrm{a}_t}{\partial h_{t-1}} = W_h^T\, \mathrm{dh}_t\,(1 - {h_t}^2)$$
$$\mathrm{d}W_x = \mathrm{dh}_t \frac{\partial \mathrm{h}_t}{\partial a_t}\frac{\partial \mathrm{a}_t}{\partial W_x} = \mathrm{dh}_t\,(1 - {h_t}^2)\, x_t^T$$
$$\mathrm{d}W_h = \mathrm{dh}_t \frac{\partial \mathrm{h}_t}{\partial a_t}\frac{\partial \mathrm{a}_t}{\partial W_h} = \mathrm{dh}_t\,(1 - {h_t}^2)\, h_{t-1}^T$$
$$\mathrm{db} = \mathrm{dh}_t \frac{\partial \mathrm{h}_t}{\partial a_t}\frac{\partial \mathrm{a}_t}{\partial \mathrm{b}} = \mathrm{dh}_t\,(1 - {h_t}^2)$$

**3. RNN Forward**

To run a vanilla RNN forward on an entire sequence of data, we just loop through T vectors for each input. From timestep t = 0 towards timestep t = T - 1, in each timestep/iteration, we have $h_t,\ cache_t = rnn\_step\_forward(x_t,\ h_{t-1},\ W_x\,,\ W_h\,, b\,)$. $\mathrm{h}_0$ is initialized to a zero matrix as there's no previous h at t = 0.

**4. RNN Backward**

In RNN backward, for each parameter, its gradient is just aggregation of gradients across all positions:

$$\mathrm{d}W_x = \sum_{t=0}^{t=T-1} d(W_x)_t$$
$$\mathrm{d}W_h = \sum_{t=0}^{t=T-1} d(W_h)_t$$
$$\mathrm{db} = \sum_{t=0}^{t=T-1} db_t$$

In each timestep/iteration, we have
$dx_t,\ dh_{t-1},\ d(W_x)_t,\ d(W_h)_t,\ db_t = rnn\_step\_backward(dh_t + dh_{t+},\ cache_t)$.

**5. Sentiment Analysis**

In the forward step, we pass forward data through four layers, namely, vanilla rnn layer, temporal affine layer, average layer and finally affine layer. Loss is calculated by taking softmax and cross-entropy of the final result of forward step. In the backward step, we calculate gradients by going backward in the order of affine layer, average layer, temporal layer and finally vanilla rnn layer.

**6. Training & Inference on Small Data**

Inference is essentailly the same as loss method except that backward operation and loss calculation are not needed. By training the RNN model on a small sample of 100 training examples for a few iterations, we get training loss values as below:

(Iteration 1 / 100) loss: 0.315483
(Iteration 11 / 100) loss: 0.277323
(Iteration 21 / 100) loss: 0.243838
(Iteration 31 / 100) loss: 0.212798
(Iteration 41 / 100) loss: 0.183810
(Iteration 51 / 100) loss: 0.157254
(Iteration 61 / 100) loss: 0.133639
(Iteration 71 / 100) loss: 0.113249
(Iteration 81 / 100) loss: 0.096055
(Iteration 91 / 100) loss: 0.081789

The figure below shows the training loss history.