2.1 Step forward

$$a = x * Wx + prev\_h * Wh + b$$
$$next\_h = \tanh(a)$$

since this is batch processing, x is placed before Wx.

2.2 Step backward

$$da = dnext\_h * (1 - next\_h^2)$$
$$dx = da * Wx^T$$
$$dprev\_h = da * Wh^T$$
$$dWx = x^T * da$$
$$dWh = prev\_h^T * da$$
$$db = \sum_{i=1}^{n} da$$

db is obtained by summing all the instances of the batch of da.

3.1 Rnn forward

Rnn forward is equivalent to run step forward function T times. For each time t, the values are calculated as following:

$$a_t = x_t * Wx + h_{t-1} * Wh + b$$
$$h_t = \tanh(a_t)$$

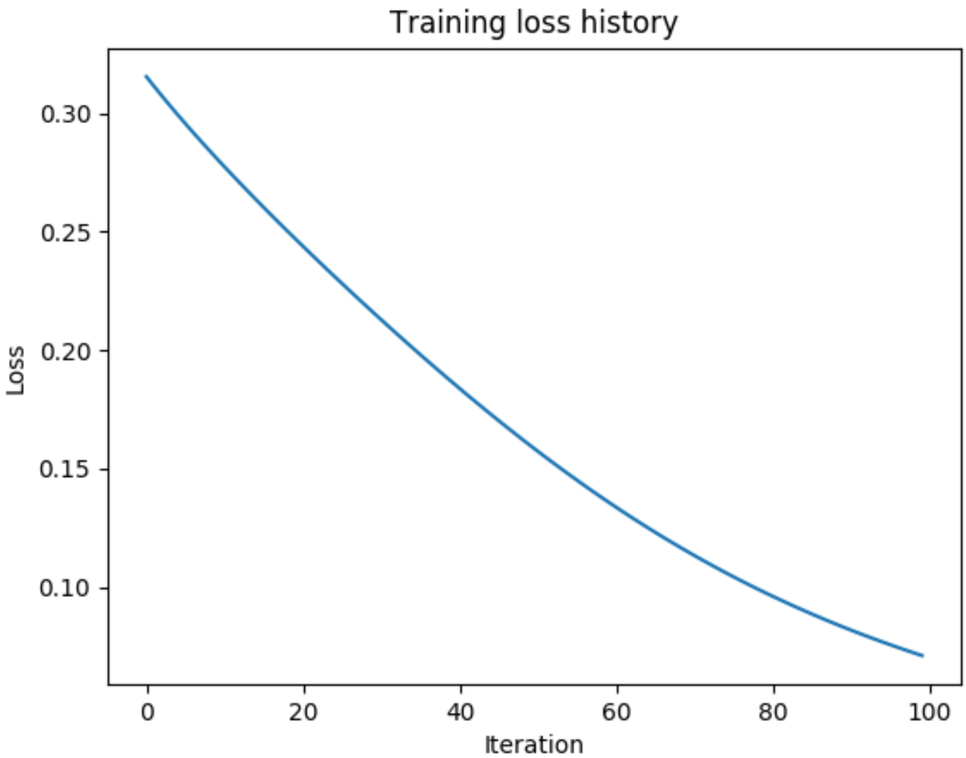
3.2 Rnn backward

$$dh\_next_{t-1} = \frac{dL}{dh_t} = \sum_{i=1}^{T} \frac{dL_i}{dh_t} = \sum_{i=t}^{T} \frac{dL_i}{dh_t} = \frac{dL_t}{dh_t} + \sum_{i=t+1}^{T} \frac{dL_i}{dh_{t+1}} \frac{dh_{t+1}}{dh_t}$$
$$= dh_t + dh\_next_t \frac{dh_{t+1}}{dh_t}$$

Here dh_next_t represents the cumulative gradients of h at timestamp t+1. dh_t is dh (input of the function) at timestamp t (it is equivalent to $\frac{dL_t}{dh_t}$).

dx, dWx, etc., can be calculated by applying $\frac{dL}{dx_t} = \frac{dL}{dh_t} \frac{dh_t}{da_t} \frac{da_t}{dx_t}$ which is same as calling step backward function twice. One with dh_t and another time with dh_next_t

5 Rnn loss



This is equivalent to stack multiple layers together.

The first layer is the RNN layer, the output rnn_out can be calculated using formula in 3.1. Rnn_out is fed into temporal affine. W is of shape [D, A], data is converted to dimension [N, T, A]. (x*W +b) with some reshaping. Output is temp_affine_out.
temp_affine_out is then fed into average forward layer. Assume stochastic gradient, mask of size (T), temp_affine_out of size (T * A). The output is calculated as
mask * temp_affine_out / (sum(mask)). With batch processing, it can be calculated with a for loop.
The output is then used to calculate softmax loss. $l(y_i, O_i) = y_i \log(O_i)$; $O_i = [\frac{\exp(v1)}{m}, \ldots]$; $m = \sum \exp(v_i)$