Backward / Forward Pass in Stack Rnns

1 Controller

At time t, controller receives input $\boldsymbol{x}_t$, previous register $\boldsymbol{r}_{t-1}$ and state $\boldsymbol{s}_{t-1}$. It outputs logits $\boldsymbol{z}_t$ for the action, state and buffer probabilities. When processed sequentually, we first embed and concat the inputs:

$$\boldsymbol{h}_t = [\text{Embed}(\boldsymbol{x}_t); \text{OneHot}(\boldsymbol{r}_{t-1}); \text{OneHot}(\boldsymbol{s}_{t-1})]$$

First we compute the action logits:

$$\boldsymbol{z}_t = \mathbf{W}_a \boldsymbol{h}_t$$

Using the hard/soft stack update functions (as defined below), we compute the new register $\boldsymbol{r}_t$. We then use this to compute the state and buffer logits:

$$\begin{aligned}
\boldsymbol{h}_{t_{new}} &= [\text{Embed}(\boldsymbol{x}_t); \text{OneHot}(\boldsymbol{r}_t); \text{OneHot}(\boldsymbol{s}_{t-1})] \\
\boldsymbol{s}_t &= \mathbf{W}_s \boldsymbol{h}_{t_{new}} \\
\boldsymbol{b}_t &= \mathbf{W}_b \boldsymbol{h}_{t_{new}}
\end{aligned}$$

The state in both the hard and soft versions is computed as $\boldsymbol{s}_t = \text{argmax}(\boldsymbol{s}_t)$. We take the buffer logit and use it directly when calculating the buffer loss.

2 Hard Stack

Stack has a pointer $\text{ptr}_t \in \mathbb{Z}$ and memory $\mathbf{M}_t \in \mathbb{R}^D$ for stack size D. Controller outputs action logits $\boldsymbol{z} \in \mathbb{R}^4$ (NOOP, PUSH_0, PUSH_1, POP).

2.1 Forward pass

model selects index $k^* = \text{argmax}(\boldsymbol{z})$. The new stack S_t is given by

$$S_{t+1} = \begin{cases} (\mathbf{M}_t, ptr_t) & \text{if } k^* = \text{NOOP} \\ (\mathbf{M}_t[\ldots], ptr_t + 1) & \text{if } k^* = \text{PUSH\_0}, \text{PUSH\_1} \\ (\mathbf{M}_t[\ldots], ptr_t - 1) & \text{if } k^* = \text{POP} \end{cases}$$

2.2 Backward pass

We want the gradient of the loss $\mathcal{L}$ wrt. the controller logits $\boldsymbol{z}$. By chain rule:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}} = \frac{\partial \mathcal{L}}{\partial S_{t+1}} \cdot \frac{\partial S_{t+1}}{\partial k^*} \cdot \underbrace{\frac{\partial k^*}{\partial \boldsymbol{z}}}_{\mathbf{0}}$$

3 Soft Stack

Let the state at time t be the matrix $\mathbf{V}_t \in \mathbb{R}^{D \times 3}$, where D is the stack depth. Row d represents the probability distribution of the symbol stored at depth d.

3.1 Forward pass

Controller outputs action probabilities $\boldsymbol{p} \in \mathbb{R}^4$ (NOOP, PUSH_0, PUSH_1, POP). The next state is computed as a weighted sum of three transformations of the current stack:

1. **NOOP:** $\mathbf{V}_{\text{hold}} = \mathbf{V}_t$
2. **Push/ Shift down:** $\mathbf{V}_{\text{down}} = \text{Roll}(\mathbf{V}_t, +1)$. The top row is overwritten with the pushed value vector $\boldsymbol{v}_{\text{push}}$ (derived from PUSH_0/PUSH_1 ratio).
3. **Pop/ Shift up:** $\mathbf{V}_{\text{up}} = \text{Roll}(\mathbf{V}_t, -1)$. The bottom row is padded with nulls.

The update is a linear combination:

$$\mathbf{V}_{t+1} = p_{\text{noop}} \cdot \mathbf{V}_{\text{hold}} + (p_{\text{push0}} + p_{\text{push1}}) \cdot \mathbf{V}_{\text{down}} + p_{\text{pop}} \cdot \mathbf{V}_{\text{up}}$$

3.2 Backward pass

We want the gradient of the loss $\mathcal{L}$ wrt. the controller logits $\boldsymbol{z}$. By chain rule:

$$\frac{\partial \mathcal{L}}{\partial z_j} = \sum_{k \in \text{Actions}} \left(\frac{\partial \mathcal{L}}{\partial \mathbf{V}_{t+1}} \cdot \frac{\partial \mathbf{V}_{t+1}}{\partial p_k} \cdot \frac{\partial p_k}{\partial z_j} \right)$$

$\frac{\partial \mathbf{V}_{t+1}}{\partial p_k}$ is the shifted matrix corresponding to action k, i.e. if $k = \text{POP}$:

$$\frac{\partial \mathbf{V}_{t+1}}{\partial p_{\text{pop}}} = \mathbf{V}_{\text{up}}$$

Here $\frac{\partial p_k}{\partial z_j}$ is given by the softmax derivative:

$$\frac{\partial p_k}{\partial z_j} = p_k(\delta_{kj} - p_j)$$