Recurrent Networks

`torch.nn.RNN`

<table>
<tr><td colspan="2"><code>torch.nn.RNN(input_size, hidden_size, num_layers=1, nonlinearity='tanh', bias=True, batch_first=False, dropout=0, bidirectional=False)(input, hidden)</code></td></tr>
<tr><td rowspan="6"></td><td>Require</td></tr>
<tr><td>• $input\_size, hidden\_size, num\_layers > 0$
• Let $num\_directions \triangleq$ <code>if</code> $bidirectional$ <code>then</code> 2 <code>else</code> 1
• $\texttt{rank}(|input|) = \texttt{rank}(|hidden|) = 3$
• If $batch\_first == False$ then
  – $|input| = (d_1, N, input\_size)$
• If $batch\_first == True$ then
  – $|input| = (N, d_2, input\_size)$
• $|hidden| = (num\_layers \times num\_directions, N, hidden\_size)$</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>• 2-tuple of (y, z) s.t.
  – $|y| = |input|[1{:}2]@(num\_directions \times hidden\_size)$
  – $|z| = |hidden|$</td></tr>
<tr><td>Comment</td></tr>
<tr><td>• 간단한 recurrent 레이어입니다.
• 입력 텐서의 브로드캐스팅 적용 안 되고 무조건 rank-3여야 합니다.
• $hidden$ 텐서 shape의 constraint는 $batch\_first$ 변수와 무관함</td></tr>
</table>

$$\frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c_e \\ \sigma \vdash H \Rightarrow h, c_h \\ c_{dim} = \{(input\_size, hidden\_size, num\_layers > 0) \land (\texttt{rank}(e) = 3) \land (\texttt{rank}(h) = 3)\} \\ num\_directions = \texttt{if}\ bidirectional\ \texttt{then}\ 2\ \texttt{else}\ 1 \\ c_{input} = \{(\texttt{if}\ batch\_first\ \texttt{then}\ (e[1] = h[2])\ \texttt{else}\ (e[2] = h[2]))\} \\ c_{hidden} = \{(h[1] = num\_layers \times num\_directions) \land (h[3] = hidden\_size)\} \\ e' = e[1{:}2]@(num\_directions \times hidden\_size) \end{array}}{\sigma \vdash \texttt{RNN}(input\_size, hidden\_size, num\_layers = 1, ..., bidirectional = False)(E, H) \Rightarrow (e', h), c_e \cup c_h \cup c_{dim} \cup c_{input} \cup c_{hidden}}$$

텐서 2개(output, changed hidden states)를 묶은 2-tuple 형태로 반환

torch.nn.LSTM

<table>
<tr><td colspan="2"><code>torch.nn.LSTM(input_size, hidden_size, num_layers=1, bias=True, batch_first=False, dropout=0, bidirectional=False)(input, (hidden, cell))</code></td></tr>
<tr><td rowspan="6"></td><td>Require</td></tr>
<tr><td>

- $input\_size, hidden\_size, num\_layers > 0$
- Let $num\_directions \triangleq$ `if` $bidirectional$ `then` 2 `else` 1
- $\texttt{rank}(|input|) = \texttt{rank}(|hidden|) = \texttt{rank}(|cell|) = 3$
- If $batch\_first == False$ then
  - $|input| = (d_1, N, input\_size)$
- If $batch\_first == True$ then
  - $|input| = (N, d_2, input\_size)$
- $|hidden| = |cell|$
$= (num\_layers \times num\_directions, N, hidden\_size)$

</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>

- 2-tuple of $(y, (z, w))$ s.t.,
  - $|y| = |input|[1{:}2]@(num\_directions \times hidden\_size)$
  - $|z| = |w| = |hidden|$

</td></tr>
<tr><td>Comment</td></tr>
<tr><td>

- Cell state도 추가적으로 가지는 LSTM 레이어입니다.
- $(hidden, cell)$을 왜 묶어서 표현했는지는 예제 코드를 보시면 명료하게 이해되실 겁니다.

</td></tr>
</table>

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c_e \\ \sigma \vdash H \Rightarrow h, c_h \\ \sigma \vdash L \Rightarrow l, c_l \quad \text{(cell)} \\ c_{dim} = \{(input\_size, hidden\_size, num\_layers > 0) \wedge (\texttt{rank}(e) = 3) \wedge (\texttt{rank}(h) = 3) \wedge (\texttt{rank}(l) = 3)\} \\ num\_directions = \texttt{if } bidirectional \texttt{ then } 2 \texttt{ else } 1 \\ c_{input} = \{(\texttt{if } batch\_first \texttt{ then } (e[1] = h[2]) \texttt{ else } (e[2] = h[2]))\} \\ c_{hidden} = \{(h[1] = num\_layers \times num\_directions) \wedge (h[3] = hidden\_size)\} \\ c_{cell} = \{(l[1] = num\_layers \times num\_directions) \wedge (l[3] = hidden\_size)\} \\ e' = e[1{:}2]@(num\_directions \times hidden\_size)\end{array}}{\sigma \vdash \texttt{LSTM}(input\_size, hidden\_size, num\_layers = 1, ..., bidirectional = False)(E, (H, L)) \Rightarrow (e', (h, l)), c_e \cup \cdots \cup c_{cell}}$$

예제 코드 참조 바랍니다.

Example Codes:

```
lstm = torch.nn.LSTM(input_size=5, hidden_size=3, num_layers=4)
input = torch.randn(7, 6, 5)  # sequence length = 7, batch size = 6
hidden = torch.randn(4, 6, 3)
cell = torch.randn(4, 6, 3)

output, (hidden_out, cell_out) = lstm(input, (hidden, cell))
print(output.shape)           # (7, 6, 3)
print(hidden_out.shape)       # (4, 6, 3)
print(cell_out.shape)         # (4, 6, 3)
```

`torch.nn.GRU`

<table>
<tr><td colspan="2"><code>torch.nn.GRU(input_size, hidden_size, num_layers=1, bias=True, batch_first=False, dropout=0, bidirectional=False)(input, hidden)</code></td></tr>
<tr><td rowspan="6"></td><td>Require</td></tr>
<tr><td>

- $input\_size, hidden\_size, num\_layers > 0$
- Let $num\_directions \triangleq$ `if` $bidirectional$ `then` 2 `else` 1
- $\texttt{rank}(|input|) = \texttt{rank}(|hidden|) = 3$
- If $batch\_first == False$ then
  - $|input| = (d_1, N, input\_size)$
- If $batch\_first == True$ then
  - $|input| = (N, d_2, input\_size)$
- $|hidden| = (num\_layers \times num\_directions, N, hidden\_size)$

</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>

- 2-tuple of (y, z) s.t.
  - $|y| = |input|[1{:}2]@(num\_directions \times hidden\_size)$
  - $|z| = |hidden|$

</td></tr>
<tr><td>Comment</td></tr>
<tr><td>

- 비교적 최근에 나온 GRU 레이어입니다.
- RNN과 shape 연산이 비슷합니다.

</td></tr>
</table>

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c_e \\ \sigma \vdash H \Rightarrow h, c_h \\ c_{dim} = \{(input\_size, hidden\_size, num\_layers > 0) \wedge (\texttt{rank}(e) = 3) \wedge (\texttt{rank}(h) = 3)\} \\ num\_directions = \texttt{if}\ bidirectional\ \texttt{then}\ 2\ \texttt{else}\ 1 \\ c_{input} = \{(\texttt{if}\ batch\_first\ \texttt{then}\ (e[1] = h[2])\ \texttt{else}\ (e[2] = h[2]))\} \\ c_{hidden} = \{(h[1] = num\_layers \times num\_directions) \wedge (h[3] = hidden\_size)\} \\ e' = e[1{:}2]@(num\_directions \times hidden\_size)\end{array}}{\sigma \vdash \texttt{GRU}(input\_size, hidden\_size, num\_layers = 1, ..., bidirectional = False)(E, H) \Rightarrow (e', h), c_e \cup c_h \cup c_{dim} \cup c_{input} \cup c_{hidden}}$$

텐서 2개(output, changed hidden states)를 묶은 2-tuple 형태로 반환

Embeddings

`torch.nn.Embedding`

<table>
<tr><td colspan="2"><code>torch.nn.Embedding(num_embeddings, embedding_dim, padding_idx=None, max_norm=None, norm_type=2.0, scale_grad_by_freq=False, sparse=False, _weight=None)(input)</code></td></tr>
<tr><td rowspan="6"></td><td>Require</td></tr>
<tr><td>

- $|input| = (d_1, d_2, \ldots, d_k)$
- $\_weight$ is $None$ or $|\_weight| = (num\_embeddings, embedding\_dim)$

</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>

- $|y| = (d_1, d_2, \ldots, d_k, embedding\_dim)$

</td></tr>
<tr><td>Comment</td></tr>
<tr><td>

- Input dimension이 너무 큰 경우를 효율적으로 처리하기 위한 row embedding
- $input$의 각 원소가 $num_embeddings$보다 작아야한다는 constraint가 있으나 shape 연산에는 영향을 주지 않음

</td></tr>
</table>

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ \sigma \vdash \_weight \Rightarrow w, c_w \quad \text{if } \_weight \neq None \\ c_w = \{(\_weight = None \vee w = (num\_embeddings, embedding\_dim))\} \\ e' = e@(embedding\_dim)\end{array}}{\sigma \vdash \texttt{Embedding}(num\_embedding, embedding\_dim, ..., \_weight = None)(E) \Rightarrow e', c \cup c_w}$$

`torch.nn.EmbeddingBag`

<table>
<tr><td colspan="2"><code>torch.nn.EmbeddingBag(num_embeddings, embedding_dim, max_norm=None, norm_type=2.0, scale_grad_by_freq=False, mode='mean', sparse=False, _weight=None, include_last_offset=False)(input, offset=None)</code></td></tr>
<tr><td rowspan="6"></td><td>Require</td></tr>
<tr><td>

- $\texttt{rank}(|input|) \in \{1, 2\}$
- If $\texttt{rank}(|input|) = 1$ then
  - $offset \neq None$ and $\texttt{rank}(|offset|) = 1$
- If $\texttt{rank}(|input|) = 2$ then
  - $offset = None$
- $\_weight$ is $None$ or $|\_weight| = (num\_embeddings, embedding\_dim)$

</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>

- If $\texttt{rank}(|input|) = 1$ then
  - $|y| = (|offset|[1], embedding\_dim)$
- If $\texttt{rank}(|input|) = 2$ then
  - $|y| = (|input|[1], embedding\_dim)$

</td></tr>
<tr><td>Comment</td></tr>
<tr><td>

- 여러 sementic constraint가 있지만 shape에는 영향을 안 줌
- $input$ 텐서의 차원에 따라 기능이 달라지는 함수

</td></tr>
</table>

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ \sigma \vdash \_weight \Rightarrow w, c_w \quad \text{if } \_weight \neq None \\ \sigma \vdash offset \Rightarrow o, c_o \quad \text{if } offset \neq None \\ c_w = \{(\_weight = None \vee w = (num\_embeddings, embedding\_dim))\} \\ c_{dim} = \{(\texttt{rank}(e) \in \{1,2\}) \wedge (\texttt{if rank}(e) = 1 \texttt{ then } (offset \neq None \wedge \texttt{rank}(o) = 1) \texttt{ else } (offset = None))\} \\ e' = \texttt{if rank}(e) = 1 \texttt{ then } (o[1], embedding\_dim) \texttt{ else } (e[1], embedding\_dim)\end{array}}{\sigma \vdash \texttt{EmbeddingBag}(num\_embedding, embedding\_dim, ..., include\_last\_offset = False)(E, offset = None) \Rightarrow e', c \cup c_w \cup c_{dim}}$$

(Builtin) `torch.nn.functional.embedding`

<table>
<tr><td colspan="2"><code>torch.nn.functional.embedding(input, weight, padding_idx=None, max_norm=None, norm_type=2.0, scale_grad_by_freq=False, sparse=False)</code></td></tr>
<tr><td rowspan="4"></td><td>Require</td></tr>
<tr><td>

- $|input| = (d_1, d_2, \ldots, d_k)$
- $|weight| = (w_1, w_2)$

</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>

- $|y| = (d_1, d_2, \ldots, d_k, w_2)$

</td></tr>
</table>

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c_e \\ \sigma \vdash W \Rightarrow w, c_w \\ c_{dim} = \{(\texttt{rank}(w) = 2)\} \\ e' = e@(w[2])\end{array}}{\sigma \vdash \texttt{embedding}(E, W, padding\_idx = None, ..., sparse = False) \Rightarrow e', c_e \cup c_w \cup c_{dim}}$$