Long sequences in TensorFlow/Keras

Dealing with something as "simple" as sequences can be surprisingly difficult in TensorFlow/Keras.

- One is required to manually break up long sequences into multiple, shorter subsequences
- The ordering of the examples in a mini-batch now becomes relevant

Consider a long sequence $\mathbf{x}^{(i)}$ of length T.

The "natural" way to represent this \boldsymbol{X} is

$$\mathbf{X} = egin{pmatrix} \mathbf{x}_{(1)}^{(1)} & \mathbf{x}_{(2)}^{(1)} & \dots & \mathbf{x}_{(T^{(1)})}^{(1)} \ \mathbf{x}_{(1)}^{(2)} & \mathbf{x}_{(2)}^{(2)} & \dots & \mathbf{x}_{(T^{(2)})}^{(2)} \ dots & dots \end{pmatrix}$$

for equal example sequence lengths $T=T^{(1)}=T^{(2)}\dots$

Suppose that the example sequence lengths T is too long (e.g., exhausts resources)

In that case, each example needs to be broken into $\mathit{multiple}$ "child-examples" of shorter length T'.

There will be T/T^\prime such child examples, each having a subsequence of length T^\prime of the parent example's sequence.

We write $\mathbf{x}^{(i,\alpha)}$ to denote child example number α of parent examples i.

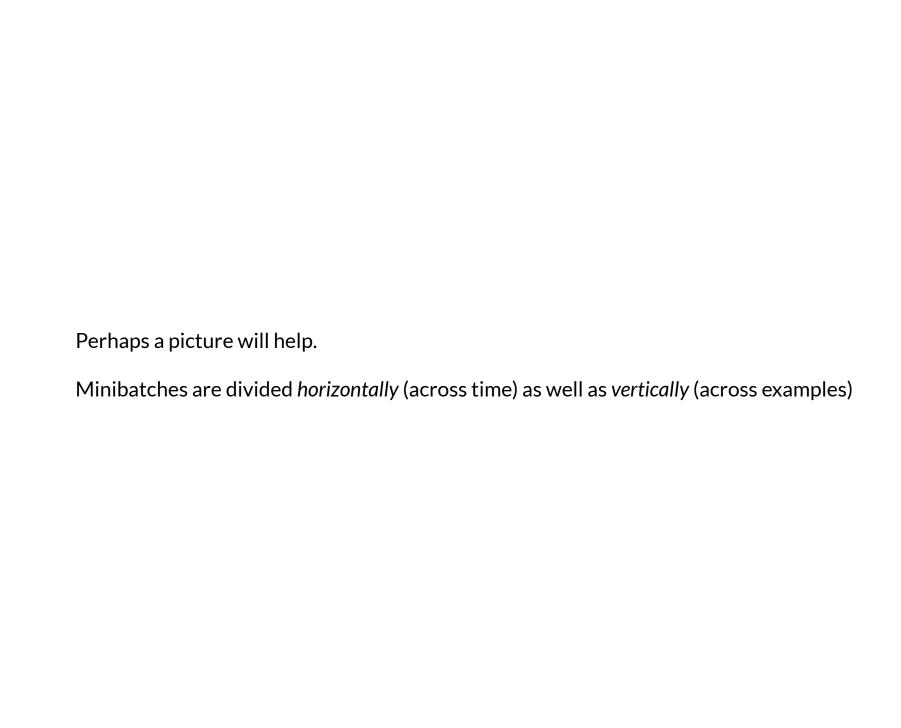
- The elements of $\mathbf{x}_{(t)}^{(i,\alpha)}$ are $\left[\begin{array}{c}\mathbf{x}_{(t)}^{(\mathbf{i})} \mid ((\alpha-1)*T/T')+1 \leq t \leq (\alpha*T/T')\end{array}\right]$.
 The subsequence $\mathbf{x}_{(t)}^{(i,\alpha+1)}$ starts right after the end of subsequence $\mathbf{x}_{(t)}^{(i,\alpha)}$

Great care must be taken when arranging child examples into a new training set \mathbf{X}' . This is because of the relationship between examples that TensorFlow implements (as of the time of this writing)

- Examples within a mini batch are considered independent
 - May be evaluated in parallel
 - So not suitable to place two children of the same parent in the same mini batch
- ullet Example i of consecutive mini-batches can be made dependent
 - With an optional flag

To get adjacent subsequences of one sequence to be treated in the proper order by TensorFlow:

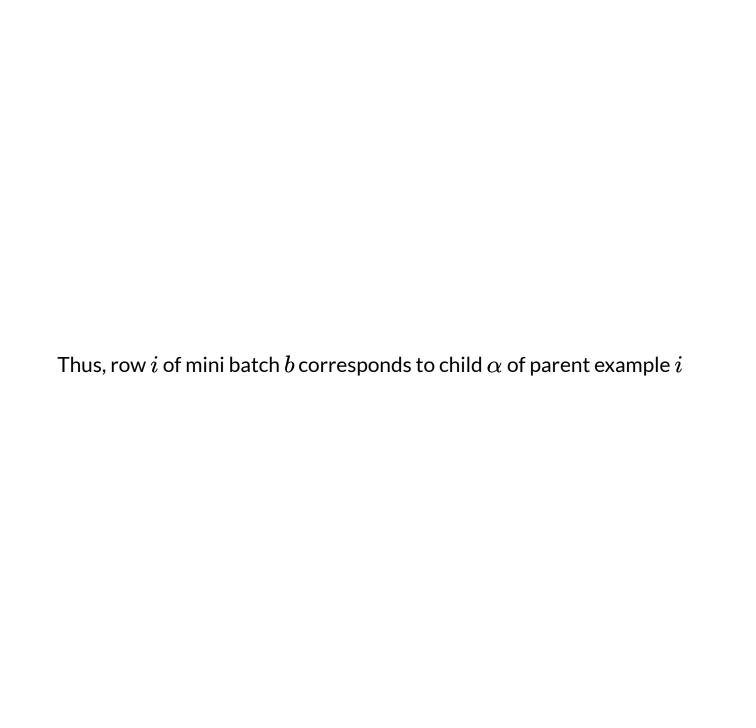
- ullet Define the number of mini batches to be T/T^\prime , which is the number of subsequences
- ullet Each subsequence of example i should be at the same position within each of the n/n' mini batches
- Set RNN optional parameter stateful=True
- When fitting the model: set shuffle=False



$$egin{aligned} ext{Minibatch 1} &= egin{pmatrix} \mathbf{x}_{(1)}^{(1)} & \mathbf{x}_{(2)}^{(1)} & \dots & \mathbf{x}_{(T')}^{(1)} \ \mathbf{x}_{(1)}^{(2)} & \mathbf{x}_{(2)}^{(2)} & \dots & \mathbf{x}_{(T')}^{(2)} \ & dots & \end{pmatrix} & ext{Minibatch 2} \ &= egin{pmatrix} \mathbf{x}_{(T'+1)}^{(1)} & \mathbf{x}_{(T'+2)}^{(1)} & \dots & \mathbf{x}_{(T'+T')}^{(1)} \ \mathbf{x}_{(T'+1)}^{(2)} & \mathbf{x}_{(T'+2)}^{(2)} & \dots & \mathbf{x}_{(T'+T')}^{(2)} \ & dots & \end{pmatrix} \end{aligned}$$

rather than

$$\mathbf{X} = \left(egin{array}{cccc} \mathbf{x}_{(1)}^{(1)} & \mathbf{x}_{(2)}^{(1)} & \dots & \mathbf{x}_{(T^{(1)})}^{(1)} \ \mathbf{x}_{(1)}^{(2)} & \mathbf{x}_{(2)}^{(2)} & \dots & \mathbf{x}_{(T^{(2)})}^{(2)} \ dots & dots \end{array}
ight)$$



Why does this work?

The flag stateful=True

- Tells TensorFlow to **not** reset the latent state of the RNN at the start of a new mini batch
 - When examples across batches are independent, the RNN should begin from step 1
 - And therefore re-initialize the latent state

By arranging the mini batches as we have

- ullet The latent state of the RNN when processing child (lpha+1) of example i
- $\bullet\,$ Is the latent state of the RNN after having process the subsequence of child α of example i

The flag shuffle=False

- Tells TensorFlow to **not** shuffle the examples in the mini batches
- $\bullet\,$ In order to preserve the fact that row i of each mini batch is a different child of the same parent

Conclusion

Long sequences present some technical issues in Keras and other frameworks.

We recognize that mitigating the issues was a highly technical topic that might take some effort to absorb.

We hope that, eventually, a better API might alleviate the burden for the end user.

```
In [3]: print("Done")
```

Done