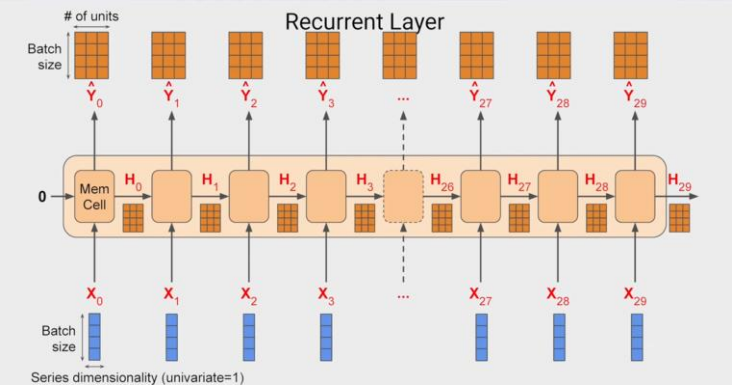
DeepLearning.AI Tensorflow Developer

Sequences, Time Series and Prediction: Week 3

Time Series RNN Model

Earlier data of time series forecasting can have impact on the later data for prediction which can be captured by LSTM model. Also, bigger window size gives better performance in this case.

RNN model input shape: (Batch Size, Window Size, One/Multiple)



RNN layers stacked on each other requires return_sequences parameter set to TRUE for returning sequence from previous layer.

Keras lambda layer API provides functionality of adding custom logic on NN data.

In order to observe impact of learning, we apply LearningRateScheduler as callback while training the model and pick Huber function for calculating loss of the model as it is less sensitive to outliers in data than MSE loss function.

```
model = tf.keras.models.Sequential([
  tf.keras.Input(shape=(window_size,1)),
  tf.keras.layers.SimpleRNN(40, return_sequences=True),
  tf.keras.layers.SimpleRNN(40),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 100.0)])

lr_schedule = tf.keras.callbacks.LearningRateScheduler(lambda epoch: 1e-8 * 10**(epoch / 20))
optimizer = tf.keras.optimizers.SGD(momentum=0.9)
model_tune.compile(loss=tf.keras.losses.Huber(), optimizer=optimizer)
history = model_tune.fit(dataset, epochs=100, callbacks=[lr_schedule])
```

Time Series LSTM Model

LSTM layer includes cell state which propagates earlier time series data to later one for prediction. LSTM layer can be wrapped around the keras bidirectional layer. Higher number of LSTM layer slightly improves the MAE error.

```
model = tf.keras.models.Sequential([
  tf.keras.Input(shape=(window_size, 1)),
  tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32, return_sequences=True)),
  tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 100.0)])
```