Machine Learning, méthodes et solutions

RNN avec Keras

RNN avec Keras

```
model = tf.keras.Sequential()
model.add(layers.Embedding(input_dim=1000, output_dim=64))
model.add(layers.SimpleRNN(128))
model.add(layers.Dense(10, activation='softmax'))
model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, None, 64)          64000
_________________________________________________________________
simple_rnn (SimpleRNN)       (None, 128)               24704
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 89,994
Trainable params: 89,994
Non-trainable params: 0
```

LSTM avec Keras

```
model = tf.keras.Sequential()
model.add(layers.Embedding(input_dim=1000, output_dim=64))
model.add(layers.LSTM(128))
model.add(layers.Dense(10, activation='softmax'))
model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, None, 64)          64000
_________________________________________________________________
lstm_1 (LSTM)                (None, 128)               98816
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 164,106
Trainable params: 164,106
Non-trainable params: 0
```

GRU avec Keras

```
model = tf.keras.Sequential()
model.add(layers.Embedding(input_dim=1000, output_dim=64))
model.add(layers.GRU(128))
model.add(layers.Dense(10, activation='softmax'))
model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, None, 64)          64000
_________________________________________________________________
gru (GRU)                    (None, 128)               74112
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1290
=================================================================
Total params: 139,402
Trainable params: 139,402
Non-trainable params: 0
```

Deep Encoder avec Keras

On peut demander à un layer récurrent à de fournir une output pour chaque timestep :

```
model = tf.keras.Sequential()
model.add(layers.Embedding(input_dim=1000, output_dim=64))
model.add(layers.GRU(256, return_sequences=True))
model.add(layers.SimpleRNN(128))
model.add(layers.Dense(10, activation='softmax'))
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, None, 64)          64000
_________________________________________________________________
gru (GRU)                    (None, 128)               74112
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1290
=================================================================
```