2017年11月22日 11:09

LSTM代码段解读

Soucrce Code:

```
def lstm(X):
        batch_size=tf.shape(X)[0]
        time_step=tf.shape(X)[1]
        w_in=weights['in']
        b_in=biases['in']
        input=tf.reshape(X,[-1,input_size])
        input_rnn=tf.matmul(input,w_in)+b_in
        input_rnn=tf.reshape(input_rnn,[-1,time_step,rnn_unit])
        cell=tf.contrib.rnn.BasicLSTMCell(rnn_unit,reuse=tf.get_variable_scope().reuse)
        multi_cell=tf.contrib.rnn.MultiRNNCell([cell for _ in range(layer_num)], state_is_tuple=True)
        init_state=multi_cell.zero_state(batch_size,dtype=tf.float32)
        output_rnn,final_states=tf.nn.dynamic_rnn(multi_cell, input_rnn,initial_state=init_state, dtype=tf.float32)
        output=tf.reshape(output_rnn,[-1,rnn_unit])
        w_out=weights['out']
        b_out=biases['out']
        pred=tf.matmul(output,w_out)+b_out
        return pred,final_states
```

来自 <https://github.com/TangYuan-Liu/RNN/blob/master/Tutorial-Learning/Stock%20Price%20Predict/N-D%20input/stock_price_predict.py>

1. For the "Tensor X", its' shape is [batch_size, time_step, input_size]
2. For the input, we reshape it to 2-D tensor, which shape is [batch_size*time_step, input_size]
3. Then we matmul the two matrix, [batch_size*time_step,input_size]*[input_size, rnn_unit], the final is [batch_size*time_step, rnn_unit]
4. Next, we reshape the final to [batch_size, time_step, rnn_unit]
5. Then we put[batch_size, time_step, rnn_unit] into the LSTMs, the output has the same shape[batch_size, time_step, rnn_unit]
6. Next, we reshape the output to[batch_size*time_step, rnn_unit], and then matmul with the matrix[rnn_unit, 1], so we will get [batch_size*time_step, 1]
7. Finally, we return the output.

So this lstm can change the input to output size from[time_step, input_size] to [time_step, 1], it means that if we want to predict a sequence which length is "time_step", the input length should be "time_step".