





莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

Tensorflow

 Python基础 ▼

 机器学习 ▼

 数据处理 ▼

 其他 ▼

莫烦PYTHON RNN 教程 ▼ 关于我 (赞助) 大家说 (神经网络 教学教程tutorial)



切换到 优酷 视频 (如优酷播放出现问题, 请 [点击这里](#))

莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

« 上一个

下一个 »

RNN LSTM 循环神经网络 (分类例子)

作者: Morvan 编辑: Morvan

- 学习资料:
 - [相关代码](#)
 - 机器学习-简介系列 [什么是RNN](#)
 - 机器学习-简介系列 [什么是LSTM RNN](#)
 - 本代码基于网上这一份代码 [code](#)

本节的内容包括:

- [设置 RNN 的参数](#)
- [定义 RNN 的主体结构](#)
- [训练 RNN](#)

莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

设置 RNN 的参数

这次我们会使用 RNN 来进行分类的训练 (Classification). 会继续使用到手写数字 MNIST 数据集. 让 RNN 从每张图片的第一行像素读到最后一行, 然后再进行分类判断. 接下来我们导入 MNIST 数据并确定 RNN 的各种参数 (hyper-parameters):

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
tf.set_random_seed(1) # set random seed

# 导入数据
mnist = input_data.read_data_sets('MNIST_data', one_hot=True)

# hyperparameters
lr = 0.001 # learning rate
training_iters = 100000 # train step 上限
batch_size = 128
n_inputs = 28 # MNIST data input (img shape: 28*28)
```

莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

```
n_classes = 10 # MNIST classes (0-9 digits)
```

接着定义 `x`, `y` 的 `placeholder` 和 `weights`, `biases` 的初始状况.

```
# x y placeholder
x = tf.placeholder(tf.float32, [None, n_steps, n_inputs])
y = tf.placeholder(tf.float32, [None, n_classes])

# 对 weights biases 初始值的定义
weights = {
    # shape (28, 128)
    'in': tf.Variable(tf.random_normal([n_inputs, n_hidden_units])),
    # shape (128, 10)
    'out': tf.Variable(tf.random_normal([n_hidden_units, n_classes]))
}
biases = {
    # shape (128, )
    'in': tf.Variable(tf.constant(0.1, shape=[n_hidden_units, ])),
    # shape (10, )
    'out': tf.Variable(tf.constant(0.1, shape=[n_classes, ]))
}
```

莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

定义 RNN 的主体结构

接着开始定义 RNN 主体结构, 这个 RNN 总共有 3 个组成部分 (`input_layer`, `cell`, `output_layer`). 首先我们先定义 `input_layer`:

```
def RNN(X, weights, biases):  
    # 原始的 X 是 3 维数据, 我们需要把它变成 2 维数据才能使用 weights 的矩阵乘法  
    # X ==> (128 batches * 28 steps, 28 inputs)  
    X = tf.reshape(X, [-1, n_inputs])  
  
    # X_in = W*X + b  
    X_in = tf.matmul(X, weights['in']) + biases['in']  
    # X_in ==> (128 batches, 28 steps, 128 hidden) 换回3维  
    X_in = tf.reshape(X_in, [-1, n_steps, n_hidden_units])
```

接着是 `cell` 中的计算, 有两种途径:

1. 使用 `tf.nn.rnn(cell, inputs)` (不推荐原因). 但是如果使用这种方法, 可以参考[这个代码](#);
2. 使用 `tf.nn.dynamic_rnn(cell, inputs)` (推荐). 这次的练习将使用这种方式.

莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

```
# 使用 basic LSTM Cell.  
lstm_cell = tf.nn.rnn_cell.BasicLSTMCell(n_hidden_units, forget_bias=1.0, state_is_tuple=True)  
init_state = lstm_cell.zero_state(batch_size, dtype=tf.float32) # 初始化全零 state
```

如果使用 `tf.nn.dynamic_rnn(cell, inputs)`, 我们要确定 `inputs` 的格式. `tf.nn.dynamic_rnn` 中的 `time_major` 参数会针对不同 `inputs` 格式有不同的值.

1. 如果 `inputs` 为 `(batches, steps, inputs)` ==> `time_major=False` ;
2. 如果 `inputs` 为 `(steps, batches, inputs)` ==> `time_major=True` ;

```
outputs, final_state = tf.nn.dynamic_rnn(lstm_cell, X_in, initial_state=init_state, time_major=False)
```

最后是 `output_layer` 和 `return` 的值. 因为这个例子的特殊性, 有两种方法可以求得 `results`.

方式一: 直接调用 `final_state` 中的 `h_state` (`final_state[1]`) 来进行运算:

```
results = tf.matmul(final_state[1], weights['out']) + biases['out']
```

方式二: 调用最后一个 `outputs` (在这个例子中,和上面的 `final_state[1]` 是一样的):

```
# 把 outputs 变成 列表 [(batch, outputs)..] * steps
```

莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

在 `def RNN()` 的最后输出 `result`

```
return results
```

定义好了 RNN 主体结构后, 我们就可以来计算 `cost` 和 `train_op`:

```
pred = RNN(x, weights, biases)
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(pred, y))
train_op = tf.train.AdamOptimizer(lr).minimize(cost)
```

训练 RNN

训练时, 不断输出 `accuracy`, 观看结果:

莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

```
accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

# init= tf.initialize_all_variables() # tf 马上就要废弃这种写法
# 替换成下面的写法:
init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    step = 0
    while step * batch_size < training_iters:
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
        batch_xs = batch_xs.reshape([batch_size, n_steps, n_inputs])
        sess.run([train_op], feed_dict={
            x: batch_xs,
            y: batch_ys,
        })
        if step % 20 == 0:
            print(sess.run(accuracy, feed_dict={
                x: batch_xs,
                y: batch_ys,
            }))
        step += 1
```

最终 **accuracy** 的结果如下:

莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

0.65625
0.726562
0.757812
0.820312
0.796875
0.859375
0.921875
0.921875
0.898438
0.828125
0.890625
0.9375
0.921875
0.9375
0.929688
0.953125
....

如果你觉得这篇文章或视频对你的学习很有帮助, 请你也分享它, 让它能再次帮助到更多的需要学习的人.

莫烦没有正式的经济来源, 如果你也想支持 莫烦Python 并看到更好的教学内容, 请拉倒屏幕最下方, 赞助他一点点, 作为鼓励他继续开源的动力.

莫烦PYTHON

« 上一个

教程 ▼

关于我


赞助

大家说

下一个 »

莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

使用社交网站账户登录

或使用来必力便捷评论 

邮件

写评论

总评论数 24

按时间正序



向阳小豆芽 2017年3月7日

请问一下，我一运行mnist = input_data.read_data_sets(`MNIST_data`, one_hot=True)就报如下错，是因为没有连上网吗？
/usr/local/Cellar/python/2.7.13/Frameworks/Python.framework/Versions/2.7/bin/python2.7
/Users/li/PycharmProjects/test/test1.py
Traceback (most recent call last):
File `~/Users/li/PycharmProjects/test/test1.py`, line 36, in
mnist = input_data.read_data_sets(`MNIST_data`, one_hot=True)
File `~/usr/local/lib/python2.7/site-packages/tensorflow/contrib/learn/python/learn/datasets/mnist.py`, line 211, in
read_data_sets
SOURCE_URL = TRAIN_IMAGES`

翻看评论

莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

支持 让教学变得更优秀

点我 赞助 莫烦

关注我的动向:

[Youtube频道](#) [优酷频道](#) [Github](#) [微博](#)

Email: morvanzhou@hotmail.com

© 2016 morvanzhou.github.io. All Rights Reserved