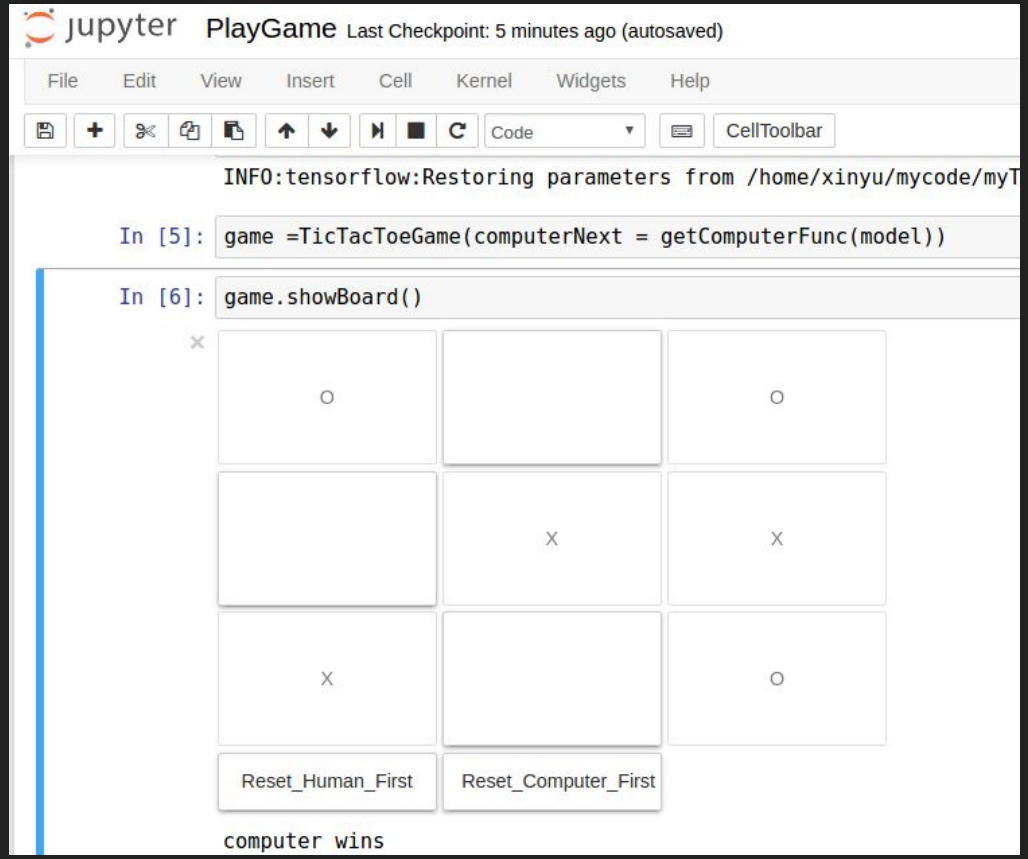
Deep learning for Tic Tac Toe

Max Liu

07/16/2017

The game board in Jupyter



Steps

1) Data preparation using random player.
2) Model training.
3) Game play.

Data preparation using random player.

No strategy at all, just randomly choose from allowed positions.

```
 6 def computerNextRandom2(humanList, computerList):
 7     """
 8
 9     Play randomly
10
11     """
12     a = [x for x in TicTacToeGame.originalList if x not in  (humanList + computerList) ]
13     if len(a) == 0:
14         return -1
15     n = random.choice(a)
16     return n
```

The data base

```
In [12]: records[:10]

Out[12]: [[8, 1, 7, 0, 2, 4, 5, '0_win'],
          [7, 5, 3, 0, 4, 6, 2, 8, 1, '0_win'],
          [8, 4, 0, 1, 5, 2, 3, 7, '1_win'],
          [3, 2, 8, 7, 0, 5, 1, 4, 6, '0_win'],
          [0, 1, 3, 5, 6, '0_win'],
          [8, 2, 3, 1, 6, 4, 0, '0_win'],
          [2, 1, 4, 7, 6, '0_win'],
          [6, 1, 8, 4, 5, 2, 7, '0_win'],
          [7, 5, 4, 2, 3, 1, 0, 6, 8, '0_win'],
          [7, 8, 5, 0, 2, 1, 3, 4, '1_win']]
```

The input and output examples

```
In [19]: inputUniq[:5]

Out[19]: [(0, -1, 1, 1, 0, -1, -1, 0, 1),
          (-1, 1, 0, -1, -1, 1, -1, 0, -1),
          (-1, 1, -1, -1, 1, 0, 0, -1, -1),
          (0, -1, 1, 1, 0, -1, 1, -1, -1),
          (-1, 1, 0, -1, 0, -1, 1, -1, -1)]

In [20]: outputs[:5]

Out[20]: [[0, 1, 0, 0, 0, 0, 0, 0, 0],
          [0, 0, 0, 1, 0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0, 0, 0, 1, 0],
          [0, 0, 0, 0, 0, 0, 0, 1, 0],
          [0, 0, 0, 0, 0, 0, 0, 1, 0]]
```

Model training

```
14 # Define the neural network
15 def build_model(dimTrainX, dimTrainY):
16     # This resets all parameters and variables, leave this here
17     tf.reset_default_graph()
18
19     # Inputs
20     net = tflearn.input_data([None, dimTrainX],dtype=tf.float32)
21
22     # Hidden layer(s)
23     net = tflearn.fully_connected(net, 10, activation='ReLU')
24
25     # Output layer and training model
26     net = tflearn.fully_connected(net, dimTrainY, activation='softmax')
27     net = tflearn.regression(net, optimizer='adam', learning_rate=0.01, loss='categorical_crossentropy')
28
29     model = tflearn.DNN(net)
30     return model
```

Test result : tf model can do better

Play 1000 times, tf model win 844 times much higher and random player (592 times) -- not too bad :)

```
In [22]: autoPlay(getComputerFunc(model),computerNextRandom2 )
Out[22]: [['0_win', 844], ['1_win', 106], ['tie', 50]]

In [23]: autoPlay(computerNextRandom2,computerNextRandom2 )
Out[23]: [['0_win', 592], ['1_win', 264], ['tie', 144]]
```

Ref:

1) https://github.com/DanielSlater/AlphaToe I wish I found this earlier :)
2) https://www.tensorflow.org/get_started/mnist/beginners This is “Hello world” for Tensorflow