УДК 004.85

ПЕРВЫЕ ШАГИ В ПОНИМАНИИ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

Муха Д.С.,
Научный руководитель – Прихожий А.А., д.т.н., профессор.

Наверное, многие, кто начинал изучение искусственных нейронных сетей, после прочтения нескольких статей задавались вопросом: «А можно полегче?». А полегче – можно.

В качестве примера создадим и обучим нейронную сеть игре в компьютерную игру CartPole. Игровая задача заключается в перемещении каретки, к которой прикреплена палка, таким образом, чтобы палка не отклонялась от вертикального положения под действием силы гравитации. Будем использовать язык программирования Python. Для взаимодействия с игрой используем библиотеку OpenAI Gym, для создания и обучения нейросети – библиотеку TFLearn.

Конфигурация сети:

1) топология – прямое распространение;
2) количество входных нейронов – 4;
3) количество выходных нейронов – 2;
4) количество слоев скрытых нейронов – 5;
5) количество нейронов в скрытых слоях – 10, 20, 30, 20, 10 соответственно на 1, 2, 3, 4, 5 слоях;
6) функция активации нейронов скрытого слоя – экспоненциальная линейная $f(x) = \begin{cases} e^x - 1, x < 0 \\ x, \quad x \geq 0 \end{cases}$;
7) функция активации выходных нейронов – Softmax, $\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$ $j = 1, \dots, K$ – применяется для слоя выходных нейронов.

```
    import tflearn
import numpy as np
import random
    import gym

network = tflearn.input_data(shape=[None, input_size, 1], name='input')
network = tflearn.fully_connected(network, 10, activation='elu')
network = tflearn.dropout(network, 0.8)
network = tflearn.fully_connected(network, 20, activation='elu')
network = tflearn.dropout(network, 0.8)
network = tflearn.fully_connected(network, 30, activation='elu')
network = tflearn.dropout(network, 0.8)
network = tflearn.fully_connected(network, 20, activation='elu')
network = tflearn.dropout(network, 0.8)
network = tflearn.fully_connected(network, 10, activation='elu')
network = tflearn.dropout(network, 0.8)
network = tflearn.fully_connected(network, 2, activation='softmax')
network = tflearn.regression(network, name='targets')
model = tflearn.DNN(network)
```

Состояние игры Cartpole характеризуется массивом из четырех чисел. В игре доступно всего два действия для исполнения – передвинуть каретку влево (0) и вправо (1). После выполнения каждого действия игра возвращает состояние, награду (всегда равна еденице), а также индикатор завершения игры, который показывает завершилась игра или нет. Максимальное количество действий в течение одного игрового эпизода составляет 200. Эпизод завершается по достижении выполнения 200 действий, или при отклонении палки на 15^{o}. Игра считается пройденной, если количество выполненных действий, усредненное по 100 запускам, больше или равно 195.

Для обучения сети необходимо составить выборку обучающих данных. Для этого будем выполнять случайные действия в игре и выбирать те действия, которые были наиболее результативными.

```
    env = gym.make('CartPole-v0')
    training_data = []
for _ in range(30000):
   observation = env.reset()
   score = 0
   game_memory = []
   while True:
      action = random.randrange(0, 2)
      game_memory.append([observation, normalize_action(action)])
      observation, reward, done, info = env.step(action)
      score += reward
      if done:
         break

   if score >= 60:
      for data in game_memory:
         training_data.append([data[0], data[1]])


def normalize_action(action):
   if action == 1:
      return [0, 1]
   else:
      return [1, 0]
```

Рассмотрим подробнее данный фрагмент кода: сперва выбираем случайное действие, далее сохраненяем состояние игры и выбранное действие, далее -выполненяем действие и прибавляем награду. Если суммарная награда по итогу игрового эпизода была достаточно высока, то можно предположить, что все действия, выбранные во время эпизода при соответствующих игровых состояниях были выбраны верно. Поэтому добавляем состояния и соответствующие им действия в результирующую выборку.

Далее обучим сеть на ранее сформированной выборке:

```
    inputs=np.array([i[0] for i in training_data]).reshape(-1,len(training_data[0][0]), 1)
targets=[i[1] for i in training_data]
model.fit({'input': inputs}, {'targets': targets})
```

Для демонстрации работы обученной сети:

```
    while True:
        env.reset()
    while True:
        env.render()
        action = np.argmax(model.predict(observation.reshape(-1, 4, 1))[0])
        observation, reward, done, _ = env.step(action)
        if done:
            env.render()
            break
```

Для сравнения приведем результаты запуска 1000 игровых эпизодов с обученной сетью, необученной сетью и результаты при выборе случайных действий:

Вариант управления игрой	количество эпизодов с суммарной наградой 200	средняя награда
обученная сеть	1000	200
необученная сеть	0	9,513
случайные действия	0	22,467

Итого созданная нейросеть сеть обучена и показывает игровые результаты, удовлетворяющие критерию успешного прохождения игры.

Литература

1. TFLearn [Электронный ресурс] / Deep learning library featuring a higher-level API for TensorFlow – Электрон. дан. – Режим доступа: http://tflearn.org/, свободный. – Загл. с экрана. – Яз. англ.
2. OpenAI Gym [Электронный ресурс] / Toolkit for developing and comparing reinforcement learning algorithms. – Электрон. дан. – Режим доступа: https://gym.openai.com/, свободный. – Загл. с экрана. – Яз. англ.