Аннотация

В данном документе приведена пояснительная записка к проекту “Глубокое обучение с подкреплением в игре Рендзю”.

В разделе “Введение” указано наименование программы на русском и английском языках и краткое описание программы.

В разделе “Назначение и область применения” указаны функциональные и эксплуатационные назначения программы, а также краткая характеристика области применения программы.

В разделе “Технические характеристики” содержатся следующие подразделы:

– постановка задачи на разработку программы;
– описание алгоритма и функционирования программы с обоснованием выбора схемы алгоритма решения задачи;
– описание и обоснование выбора метода организации входных и выходных данных;
– описание и обоснование выбора состава технических и программных средств.

Настоящий документ разработан в соответствии с требованиями:

1) ГОСТ 19.101-77 Виды программ и программных документов [1];
2) ГОСТ 19.102-77 Стадии разработки [2];
3) ГОСТ 19.103-77 Обозначения программ и программных документов [3];
4) ГОСТ 19.104-78 Основные надписи [4];
5) ГОСТ 19.105-78 Общие требования к программным документам [5];
6) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом [6];
7) ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению [7].

Изменения к Пояснительной записке оформляются согласно ГОСТ 19.603-78 [8], ГОСТ 19.604-78 [9].

Содержание

1. Введение

Наименование программы на русском языке: “Глубокое обучение с подкреплением в игре Рендзю”.

Наименование программы на английском языке: “Renju game with deep reinforcement learning”.

Краткое название программы: “Darin”. Программа предназначена для изучения игры Рендзю.

2. Назначение и область применения

2.1. Функциональное назначение

Функциональное назначение программы – предсказание следующего хода программы по данной позиции в игре

2.2. Эксплуатационное назначение и область применения

Программа будет использоваться в качестве бота для игры в Рендзю или поддержки при игре в Рендзю.

3. Технические характеристики

3.1. Постановка задачи

1) Задание на курсовую работу. Приказ 2.3-02/1501-03.«Об утверждении тем и руководителей курсовых работ студентов образовательной программы «Прикладная математика и информатика» факультета компьютерных наук» от 15.01.2019

2) Техническое задание «Глубокое обучение с подкреплением в игре Рендзю»

3) План проекта разработки

Цель разработки – создание нейронной сети, способной предсказывать следующий ход при игре в Рендзю, обученной на играх профессиональных игроков.

3.2. Описание алгоритма

Алгоритм обучения:

1. Программа преобразовывает файл с играми профессиональных игроков в файл с текущей позицией и следующим ходом. Пара «позиция-ход» добавляется в файл только в том случае, если это ход игрока, выигравшего партию.
   Текущая позиция – это три матрицы. У первой на всех координатах, соответствующих черным камням, стоят 1, на остальных клетках 0. У второй на всех координатах, соответствующих белым камням, стоят -1, на остальных клетках 0. Третья – матрица состоящая только из 1 или только из -1, зависит от очередности хода: если ход черного 1, иначе -1.
   Следующий ход – это номер класса. Каждую координату вида LETTER-NUMBER преобразовываем в класс. Соответственно классов 225.
2. Обучение нейронной сети на составленном только что датасете, где признаки – текущая позиция, метка – координаты следующего хода
3. Сохранение весов данной модели в формате checkpoint.hdf5

Алгоритм предсказания следующего хода:

1. Подгружается модель с заранее обученными весами.
2. По ней для каждой позиции того, что происходит на доске, строится предсказание следующего хода.

3.3. Описание модели

Архитектура нейронной сети: 7 Convolutonal слоев, 1 Dense слой;
Метод оптимизации: Adam
Функция потерь: categorical crossentropy loss

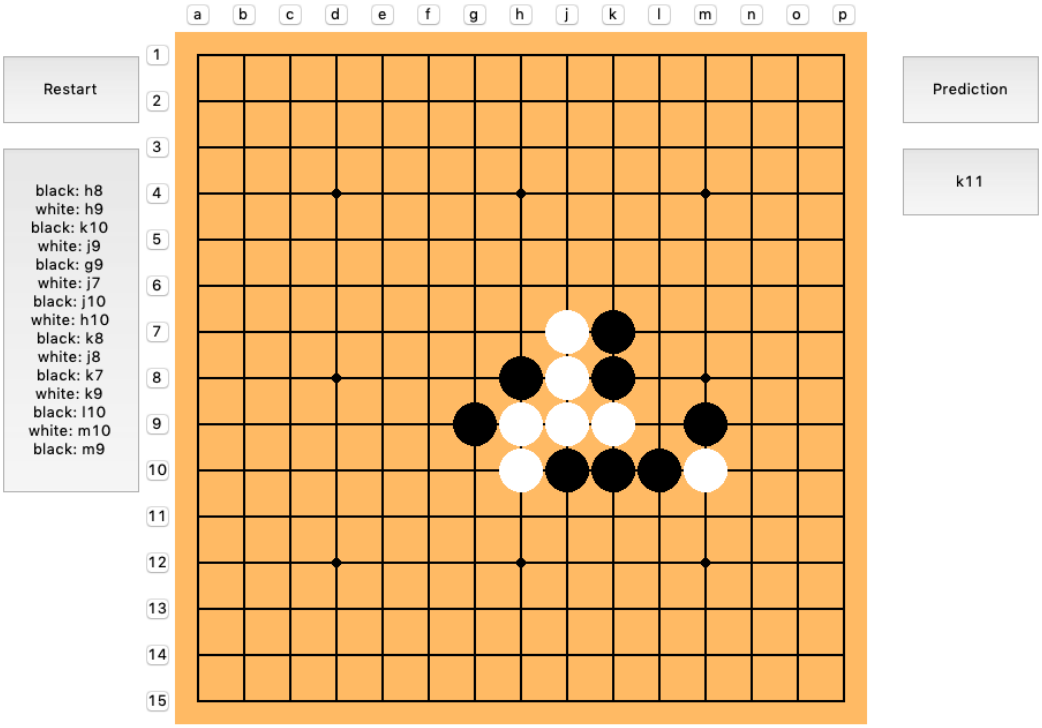
```
def make_model():
    model = Sequential()
    model.add(Conv2D(32, 3, input_shape=(15, 15, 3), padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(64, kernel_size=(5, 5), padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(128, 3, padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(64, kernel_size=(5, 5), padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(32, 3, padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(16, kernel_size=(5, 5), padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(1, 3, padding='same'))
    model.add(Activation('relu'))

    model.add(Flatten())

    model.add(Dense(225))
    model.add(Activation("softmax"))

    model.compile(optimizer=Adam(lr=1e-4), loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

```python
from tensorflow.python.keras.callbacks import CSVLogger
logger = CSVLogger('log.csv', append=True, separator=';')

INIT_LR = 8e-3
BATCH_SIZE = 1024
EPOCHS = 500

with tf.device('/device:GPU:3'):
    #model = load_model('day_model.hdf5')
    model = make_model()  # define our model
    model_checkpoint = ModelCheckpoint('day_model.hdf5', monitor='loss', verbose=1, save_best_only=True)

    history = model.fit(
        xTrain2, yTrain,  # prepared data
        batch_size=BATCH_SIZE,
        callbacks=[model_checkpoint, logger],
        epochs=EPOCHS,
        validation_data=(xVal2, yVal),
        shuffle=True,
    )
```

```
Epoch 00020: loss improved from 1.98495 to 1.96375, saving model to day_model.hdf5
Epoch 21/500
912488/912488 [==============================] - 214s 234us/step - loss: 1.9456 - acc: 0.4509 - val_loss: 1.9982 -
val_acc: 0.4382

Epoch 00021: loss improved from 1.96375 to 1.94561, saving model to day_model.hdf5
Epoch 22/500
912488/912488 [==============================] - 214s 234us/step - loss: 1.9285 - acc: 0.4541 - val_loss: 2.0043 -
val_acc: 0.4391
```

3.4. Описание, обоснование метода организации входных, выходных данных

Был разработан пользовательский интерфейс:



По нажатию на соответствующую позицию на доске появляется камень черного или белого цвета в зависимости от активного игрока. В левом верхнем углу находится кнопка перезапуска партии. В левом нижнем углу назодится история текущей игры. В правом верхнем углу – возможность предсказать ход по данному состоянию игры и результат предсказания. Есть опция показывать предсказания автоматически после каждого хода (режим игры с ботом) или по нажатию на клавишу (режим самостоятельной игры с помощью программы)

3.5. Описание, обоснование метода выбора технических средств

Рекомендуемые характеристики компьютера:

Частота процессора не менее 2.3 ГГц

Объем оперативной памяти от 8192 МБ ОЗУ

Программное обеспечение:

Python 3.6 и библиотеки: numpy, matplotlib, pickle, tensorflow 1.12.0, keras 2.2.4.

4. Технико-экономические показатели

Проект не предусматривает монетизацию. Проект не является уникальным решением, разобранные алгоритмы – достаточно хорошо изученный за последние несколько лет путь. Программа реализована в рамках выполнения программного проекта за 2 курс.

5. Список использованной литературы

1. Optimal Control Theory: Introduction. 1970 / Kirk, Donald E. Robbins, Herbert p. 527-535.
2. Machine Learning: A Probabilistic Perspective. 2012 / Mutphy, Kevin P. / p.
3. Deep Learning, An MIT Press book. 2016 / P Ian Goodfellowi, L. Gargano Eds. - Ischia. – p. 319-330.
4. Artificial Intelligence: A Modern Approach. 2010 / Stuart J. Russell, Peter Norvig. Prentice Hall. / p. 28-40
5. Reinforcement learning: An Introduction. 2014 / Richard S. Sutton, Andrew G. Barto. - pp. 1-92.
6. Mastering the game of Go without human knowledge 2014 / David Silver, Karen Simonyan

6. Описание алгоритмов и модели

Ключевые скрипты, программы, ноутбуки: preparation.ipynb, nn.ipynb, checkpoint.hdf5, UI.ipynb, darinkim.py

1. preparation.ipynb – предподготовка данных:
   1.1. translate, translate2 – преобразовывает координату вида LETTER-NUMBER в (coord1, coord2), затем в соответствующий класс.
   1.2. to_seq – преобразовывает список ходов в нужный нам вид.
2. nn.ipynb – нейронная сеть и тренировка сети:
   2.1. предподготовка данных: создание сетов train и val.
   2.2. make_model – описание модели
   2.3. обучение модели
3. UI.ipynb – пользовательский интерфейс и предсказание работы модели:
   3.1. give_pic – по клетке выдает картинку, которую должна иметь данная клетка
   3.2. win_condition – проверка на возможность завершения игры, так как игры в датасете редко заканчиваются именно пятью камнями, так что мало примеров заканчивания игры
   3.3. predictShow – предсказание следующего хода при нажатии на кнопку
   3.4. refresh – возвращение партии на стартовую позицию
4. darinkim.py – агент для участия в турнире нейронных сетей. Фактически аналог UI.ipynb, написанный под особенный формат турнира. Функции дублируют UI.ipynb без графической оболочки