УНИВЕРСИТЕТ
искусственного
интеллекта

Генетические алгоритмы для подбора параметров и архитектуры нейросетей

Генетический алгоритм и нейронные сети

Часто встречаются упоминания о том, что нейронная сеть может быть обучена каким-либо методом глобальной оптимизации, например, генетическим алгоритмом. В самом деле, такая мысль возникает достаточно легко: если возможно использовать генетический алгоритм для поиска минимума некоторой сложной функции, то точно так же можно обучить и нейронную сеть, приняв за целевую функцию функцию ошибки сети.

Проблемой здесь является то, что в отличие от традиционного подхода к обучению сетей с использованием градиентных методов, генетический алгоритм является неуправляемым алгоритмом поиска, то есть в общем случае процесс движется в произвольном направлении, в то время как при использовании градиентных методов процесс движется в сторону минимум целевой функции.

Кроме этого, в случае использования генетического алгоритма для обучения сети такой подход фактически означает создание целой популяции сетей. Что влечёт за собой дополнительные временные затраты.

Посмотрим на возможные варианты использования генетического алгоритма в нелегком деле обучения нейронной сети.

1. Перебор гиперпараметров
2. Подбор архитектуры
3. Смешанный вариант
4. Предобработка (обогащение) базы данных

Генетический алгоритм и нейронные сети

Выбор структуры нейронной сети

Зачастую под обучением нейронной сети с использованием генетического алгоритма понимается выбор архитектуры сети (количество слоев, количество нейронов в слое, функции активации и т. п.), но здесь тоже не всё так гладко.

Очевидно, что для того, чтобы решить, какая архитектура сети лучше других, нужно иметь возможность сравнивать архитектуры сетей между собой. Причем сравнивать нужно с точки зрения той задачи, которую планируется решать сетью.

Линейная архитектура

Стоит обратить внимание на структуру бота для данной задачи.

Входной слой

0 - делаем ли нормализацию

1 - размер первого свёрточного слоя

2 - ядро первого свёрточного слоя

3 - функция активации первого слоя

4 - делаем ли MaxPooling0

5 - размер MaxPooling0

Первый скрытый слой

6 - Делаем ли второй сверточный слой

7 - размер второго сверточного слоя

8 - ядро второго сверточного слоя

9 - делаем ли MaxPooling1

10 - размер MaxPooling1

11 - функция активации

Второй скрытый слой

12 - Делаем ли третий сверточный слой

13 - размер третьего сверточного слоя

Генетический алгоритм и нейронные сети

14 - ядро третьего сверточного слоя

15 - делаем ли MaxPooling2

16 - размер MaxPooling2

18 - функция активации предпоследнего слоя

Третий (предпоследний) скрытый слой

20 - делаем ли нормализацию

21 - размер полносвязного слоя

Создание сети, генетический выбор элементов параметров, проверка работы сети всё это задается тремя основными функциями.

```
#Создаём сеть (net - список параметров)
def createConvNet(net):

    model = Sequential()                    # Создаем модель
Sequential

    makeFirstNormalization = net[0]  # Делаем ли
нормализацию в начале
    firstConvSize = 2 ** net[1]       # Размер первого
свёрточного слоя
    firstConvKernel = net[2]          # Ядро первого
свёрточного слоя
    activation0 = net[3]               # Функция активации
входного слоя
    makeMaxPooling0 = net[4]          # Делаем ли maxpooling
для нулевого слоя
    maxPoolingSize0 = net[5]          # Размер MaxPooling

    makeSecondConv = net[6]           # Делаем ли второй
свёрточный слой
    secondConvSize = 2 ** net[7]      # Размер второго
свёрточного слоя
    secondConvKernel = net[8]         # Ядро второго
свёрточного слоя
    makeMaxPooling1 = net[9]          # Делаем ли MaxPooling
    maxPoolingSize1 = net[10]         # Размер MaxPooling
```

Генетический алгоритм и нейронные сети

```
    activation1 = net[11]                   # Функция активации

    makeThirdConv = net[12]                 # Делаем ли второй
свёрточный слой
    thirdConvSize = 2 ** net[13]            # Размер второго
свёрточного слоя
    thirdConvKernel = net[14]               # Ядро второго
свёрточного слоя
    makeMaxPooling2 = net[15]               # Делаем ли MaxPooling
    maxPoolingSize2 = net[16]               # Размер MaxPooling
    activation2 = net[17]                   # Функция активации

    activation3 = net[18]                   # Функция активации
для
    activation4 = net[19]                   # Функция активации
для последнего слоя

    makeSecondNormalization = net[20] # Делаем ли финальную
нормализацию
    denseSize = 2 ** net[21]                # Размер
полносвязного слоя

    activation_list = ['linear','relu','tanh','softmax','sigm
oid']

    if (makeFirstNormalization):            # Если делаем
нормализацию вначале

        # Добавляем слой BatchNormalization
        model.add(BatchNormalization(input_shape=(xLen, 1)))

        # Добавляем Conv1D-слой с firstConvSize нейронами и
ядром (firstConvKernel)
        model.add(Conv1D(firstConvSize, firstConvKernel,
activation=activation_list[activation0], padding='same'))
    else:

        # Добавляем Conv1D-слой с firstConvSize нейронами и
```

Генетический алгоритм и нейронные сети

```
ядром (firstConvKernel)
     model.add(Conv1D(firstConvSize, firstConvKernel, input_
shape=(xLen, 1), activation=activation_list[activation0],
padding='same'))

   if makeMaxPooling0:                    # Если делаем
maxpooling
     model.add(MaxPooling1D(maxPoolingSize0))


   if (makeSecondConv):                   # Если делаем второй
свёрточный слой
     # Добавляем Conv1D-слой с secondConvSize нейронами и
ядром (secondConvKernel)
     model.add(Conv1D(secondConvSize, secondConvKernel,
activation=activation_list[activation1], padding='same'))

     if (makeMaxPooling1):                # Если делаем
MaxPooling
       # Добавляем слой MaxPooling1D с размером
(maxPoolingSize)
       model.add(MaxPooling1D(pool_size=maxPoolingSize1))


   if (makeThirdConv):                    # Если делаем второй
свёрточный слой
     # Добавляем Conv1D-слой с secondConvSize нейронами и
ядром (secondConvKernel)
     model.add(Conv1D(thirdConvSize, thirdConvKernel,
activation=activation_list[activation2], padding='same'))

     if (makeMaxPooling2):                # Если делаем
MaxPooling
       # Добавляем слой MaxPooling1D с размером
(maxPoolingSize, maxPoolingSize)
       model.add(MaxPooling1D(pool_size=maxPoolingSize2))
```

Генетический алгоритм и нейронные сети

```
    if (makeSecondNormalization):      # Если делаем
финальную нормализацию
        model.add(BatchNormalization()) # Добавляем слой
BatchNormalization

    model.add(Flatten())                # Добавляем слой
Flatten

    model.add(Dense(denseSize, activation=activation_
list[activation3])) # Добавляем слой Dense с denseSize
нейронами
    model.add(Dense(1, activation=activation_
list[activation4]))          # Добавляем Dense-слой с
softmax-активацией и 10 нейронами

    return model                        # Возвращаем модель
```

```
  '''
    Функция вычисления результата работы сети
  '''

  def evaluateNet(net, ep, verb):
    val = 0
    time.time()
    model = createConvNet(net) # Создаем модель
createConvNet

# Компилируем модель
model.compile(optimizer=Adam(lr=1e-4),
                    loss='mse')

history = model.fit_generator(trainDataGen,
                      epochs=5,
                      verbose=verb,
                      validation_data=testDataGen)

val = history.history["val_loss"][-1] # Возвращаем
```

Генетический алгоритм и нейронные сети

```
точность на проверочной выборке с последней эпохи

    return val, model                                   # Возвращаем
точность

  '''
    Функция создания списка случайных параметров
  '''
  def createRandomNet():
    net = []
    net.append(random.randint(0,1)) # Делаем или нет
нормализацию
    net.append(random.randint(3,6)) # Первый свёрточный
слой от 8 до 64 нейронов
    net.append(random.randint(3,7)) # Ядро первого
свёрточного слоя от 3 до 7
    net.append(random.randint(0,4)) # Функция активации
первого слоя
    net.append(random.randint(0,1)) # Делаем ли MaxPooling
    net.append(random.randint(2,5)) # Размер MaxPooling

    net.append(random.randint(0,1)) # Сколько делаем еще
сверточных слоев
    net.append(random.randint(3,6)) # Второй свёрточный
слой от 8 до 64 нейронов
    net.append(random.randint(3,7)) # Ядро второго
свёрточного слоя от 3 до 7
    net.append(random.randint(0,1)) # Делаем ли MaxPooling
    net.append(random.randint(2,5)) # Размер MaxPooling
    net.append(random.randint(0,4)) # Функция активации
второго слоя

    net.append(random.randint(0,1)) # Сколько делаем еще
сверточных слоев
    net.append(random.randint(3,6)) # Второй свёрточный
слой от 8 до 64 нейронов
    net.append(random.randint(3,7)) # Ядро второго
```

Генетический алгоритм и нейронные сети

```
свёрточного слоя от 3 до 7
    net.append(random.randint(0,1)) # Делаем ли MaxPooling
    net.append(random.randint(2,5)) # Размер MaxPooling
    net.append(random.randint(0,4)) # Функция активации
второго слоя

    net.append(random.randint(0,4)) # Функция активации
предпоследнего dense слоя
    net.append(random.randint(0,2)) # Функция активации
последнего слоя

    net.append(random.randint(0,1)) # Делаем ли финальную
нормализацию слой
    net.append(random.randint(3,6)) # Размер полносвязного
слоя от 8 до 64

    return net
```

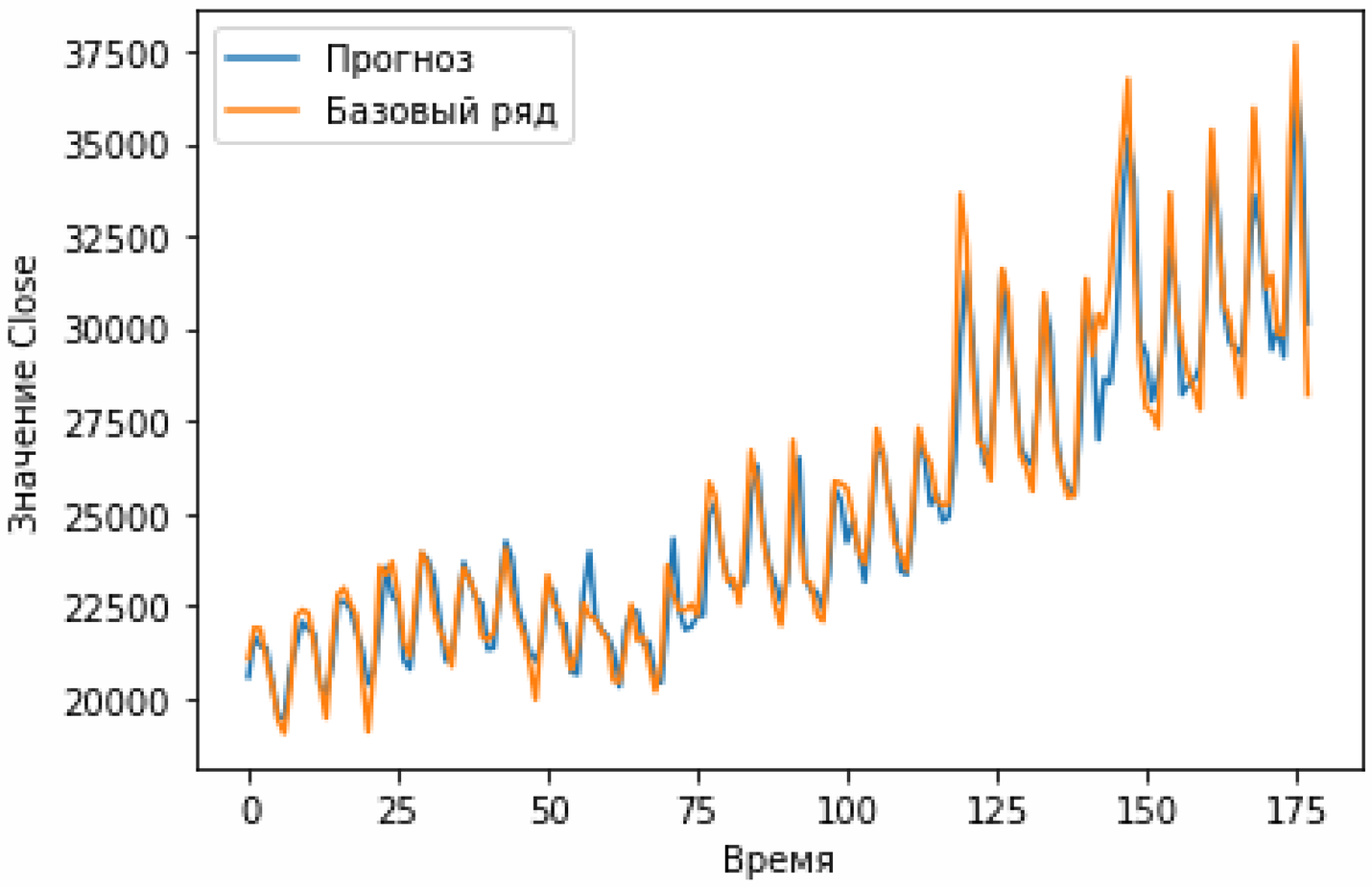
Такой набор параметров использовался для обучения:

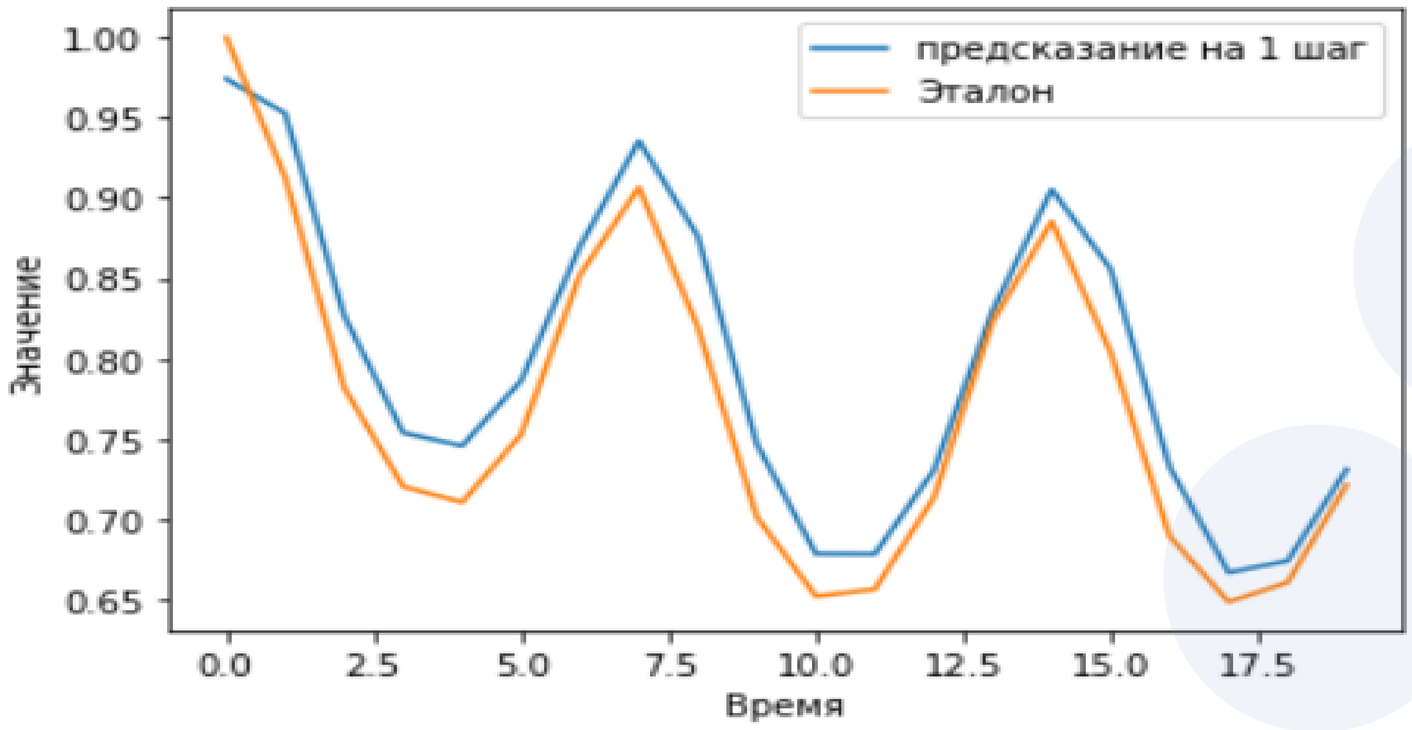
```
n = 20                  # Общее число ботов
nsurv = 10              # Количество выживших (столько лучших
переходит в новую популяцию)
nnew = n - nsurv        # Количество новых (столько новых
ботов создается)
l = 22                  # Размер бота
epohs = 10              # Количество эпох

mut = 0.09              # Коэффициент мутаций
```

Данным алгоритмом решалась задача из темы «Временные ряды» по базе данных трафика сайта.

Результат прогноза, ошибки и корреляция выведены на графиках.

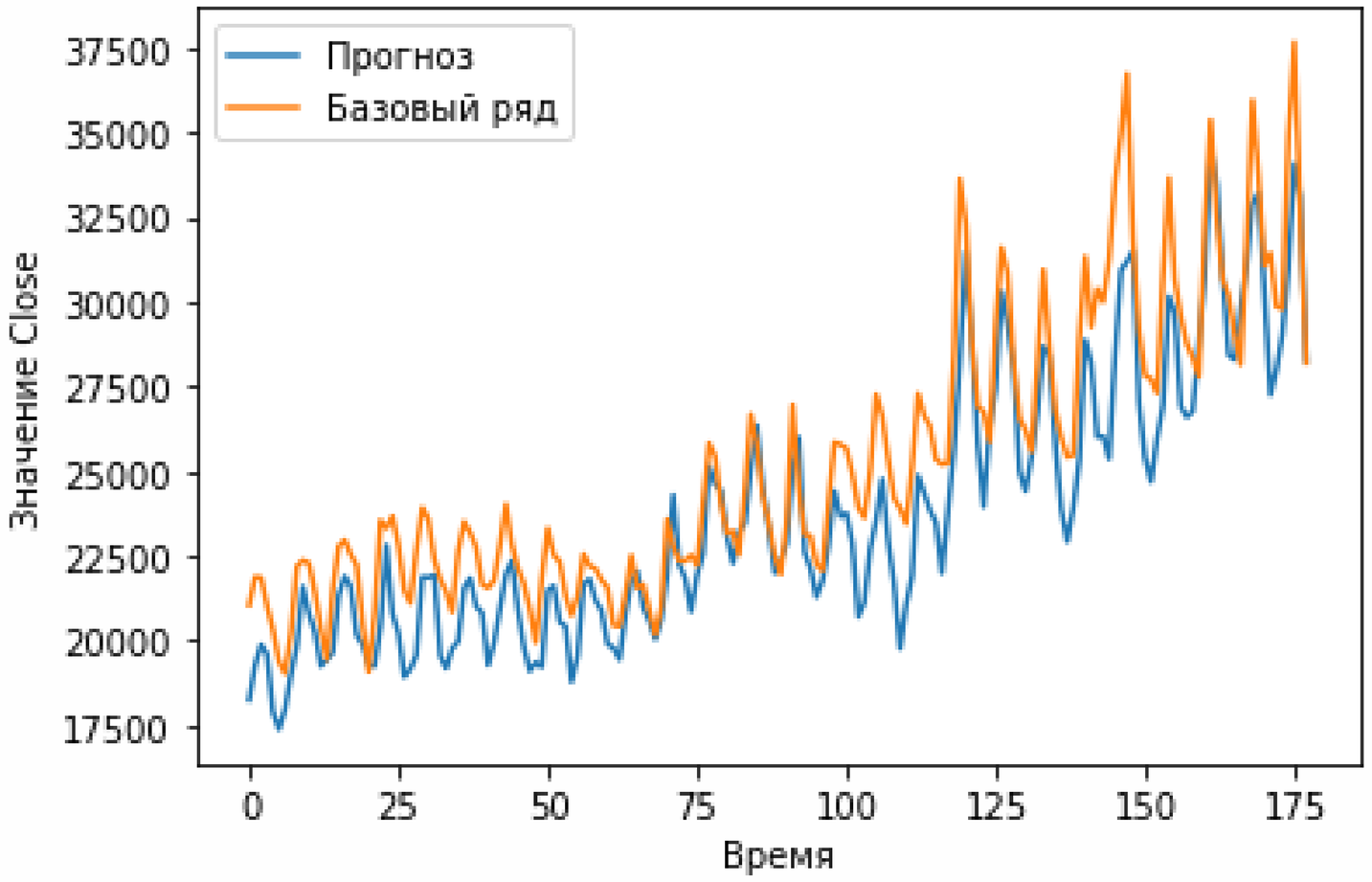
Генетический алгоритм и нейронные сети



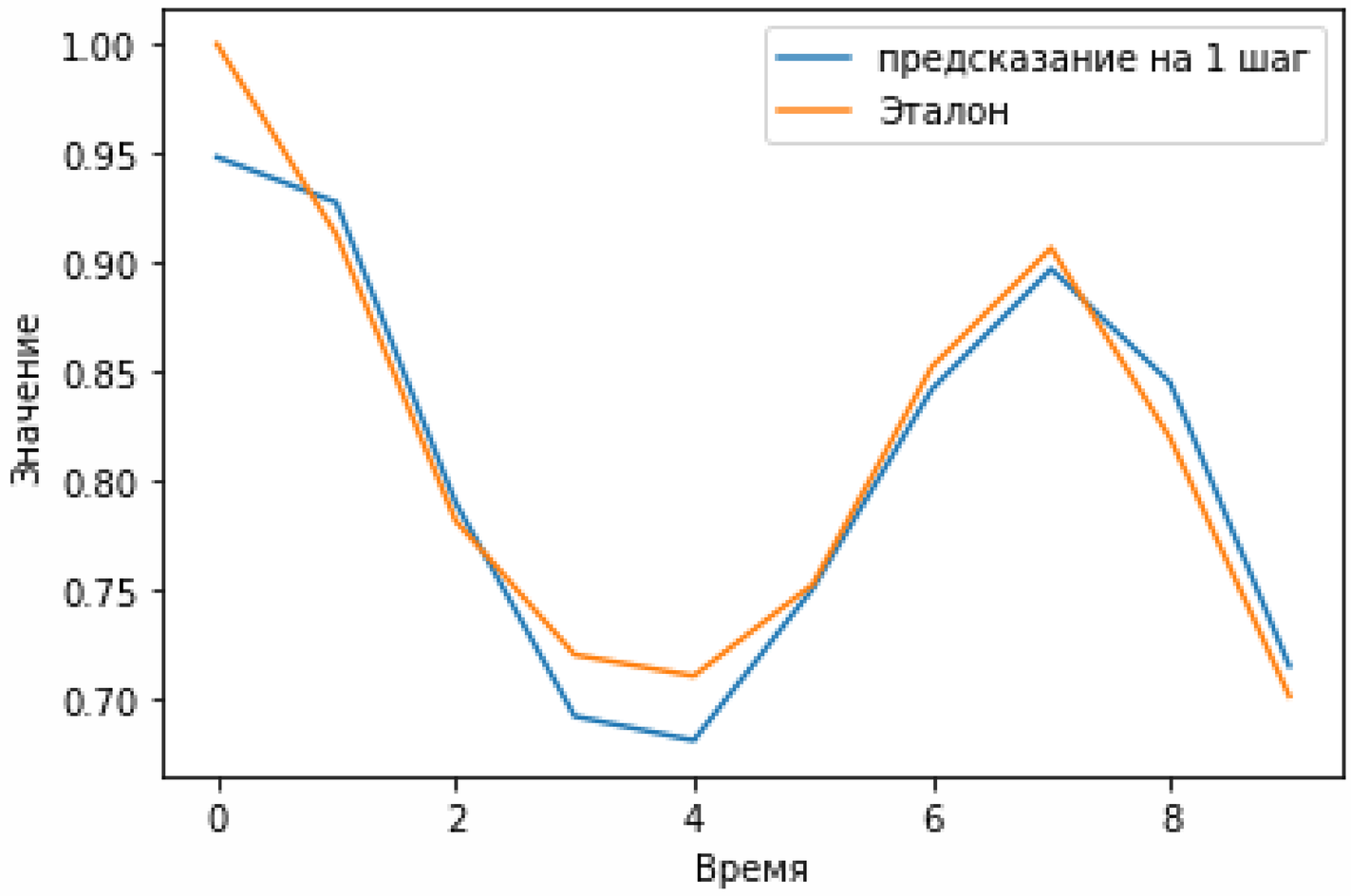


Генетический алгоритм и нейронные сети

А теперь выведем такие же графики из созданной ранее сетью в ручном режиме



Генетический алгоритм и нейронные сети



Далее были проведены эксперименты с подбором нелинейной архитектуры.

Создаем набор для бота из 27 параметров.

БЛОК 1

0 - делаем ли нормализацию

1 - размер MaxPooling для всех слоев

2 - размер первого свёрточного слоя

3 - ядро первого свёрточного слоя

4 - функция активации первого слоя

БЛОК 2

5 - делаем ли второй свёрточный слой

6 - размер второго свёрточного слоя

7 - ядро второго свёрточного слоя

8 - делаем ли MaxPooling

9 - функция активации второго слоя

БЛОК 3

10 - делаем ли второй свёрточный слой

11 - размер второго свёрточного слоя

12 - ядро второго свёрточного слоя

Генетический алгоритм и нейронные сети

13 - делаем ли MaxPooling

14 - функция активации второго слоя

БЛОК 4

15 - делаем ли второй свёрточный слой

16 - размер второго свёрточного слоя

17 - ядро второго свёрточного слоя

18 - делаем ли MaxPooling

19 - функция активации второго слоя

БЛОК 5

20 - делаем ли второй свёрточный слой

21 - размер второго свёрточного слоя

22 - ядро второго свёрточного слоя

23 - делаем ли MaxPooling

24 - функция активации второго слоя

БЛОК 6

25 - делаем ли нормализацию перед полносвязным слоем

26 - делаем ли полносвязный слой

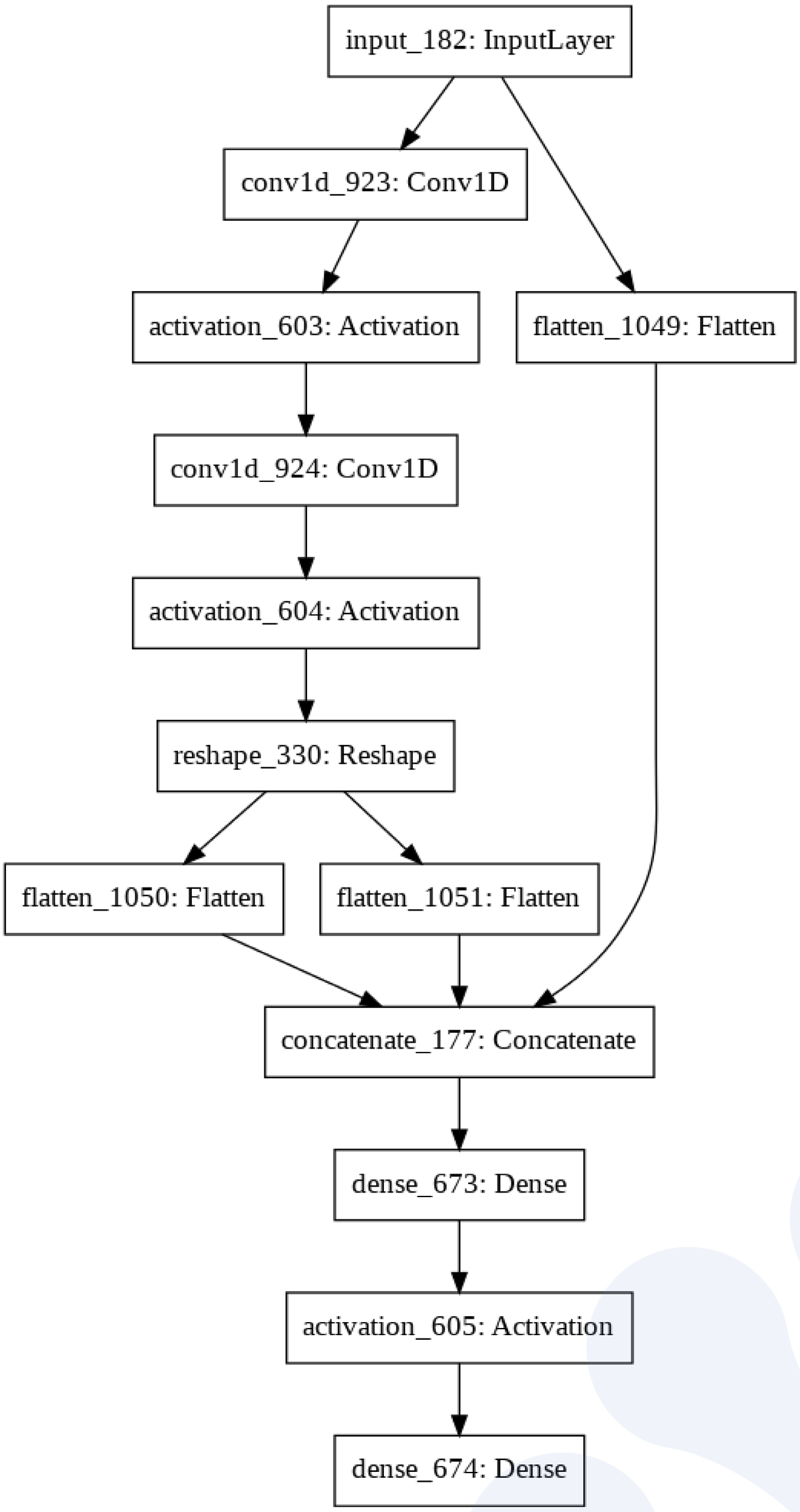
27 - размер полносвязного слоя

Проводим эволюционные испытания на параметрах:

```
n = 20                          # Общее число ботов
nsurv = 7                       # Количество выживших (столько
лучших)
nnew = n - nsurv                # Количество новых (столько
новых ботов создается)
l = 28                          # Размер бота
epohs = 9                       # Количество эпох
mut = 0.4                       # Коэффициент мутаций
```

Генетический алгоритм и нейронные сети

Генетический алгоритм выбрал лучшую популяцию и архитектура сети получилась такой:



Как видно, применение генетических алгоритмов в подборе параметров нейросетей способно расширить границы их создания и обучения.