**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. М. В. ЛОМОНОСОВА**

**ФИЗИЧЕСКИЙ ФАКУЛЬТЕТ**

**КАФЕДРА МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ И ИНФОРМАТИКИ**

**ГЕНЕРАЦИЯ ИЗОБРАЖЕНИЙ ДЛЯ АНАЛИЗА УСТОЙЧИВОСТИ НЕЙРОННЫХ СЕТЕЙ В ЗАДАЧЕ КЛАССИФИКАЦИИ**

Курсовая работа

студента 435 группы

Клиентова Григория Алексеевича

Научные руководители:

д. ф.-м. н., профессор П. В. Голубцов

Дьяконов Александр Геннадьевич

**Москва**

**2024**

Оглавление

[1. Введение 3](#_Toc167069041)

[2. Цель работы 3](#_Toc167069042)

[3. Используемая модель для атак 3](#_Toc167069043)

[4. Описание алгоритма для генерации изображений 4](#_Toc167069044)

[5. Используемые маскирования 4](#_Toc167069045)

[6. Эксперименты 5](#_Toc167069046)

[6.1 Маскирование с разбиением изображения на k равных частей 5](#_Toc167069047)

[5. Описание реализованного алгоритма 6](#_Toc167069048)

[6. Полученные результаты 6](#_Toc167069049)

[6.1 Задача аппроксимации 6](#_Toc167069050)

[6.2 Задача классификации «Исключающее ИЛИ» 11](#_Toc167069051)

[7. Основные итоги работы 14](#_Toc167069052)

[8. Литература 15](#_Toc167069053)

# Введение

В последние годы машинное обучение стало неотъемлемой частью многих областей, включая компьютерное зрение, обработку естественного языка и медицинскую диагностику. Одной из ключевых задач в этих областях является классификация изображений, которая позволяет компьютерам автоматически определять содержание изображений. Однако, несмотря на значительные успехи в области классификации изображений, машинное обучение остается уязвимым для атак, известных как атаки с использованием adversarial examples.

Adversarial examples - это специально созданные входные данные, которые предназначены для обмана моделей машинного обучения. Такие примеры могут привести к тому, что модель выдаст неверный результат, даже если исходное изображение легко распознается человеком. Эта проблема стала предметом активных исследований, поскольку она ставит под сомнение надежность и безопасность систем машинного обучения.

В данной работе исследуется вариант adversarial attack, когда на вход сети подается некоторое универсальное изображение с примененным маскированием, которое модель будет относить к любому заданному классу, причем ответ модели будет зависеть от типа и расположения маски.

# 2. Цель работы

* Реализовать алгоритм генерации универсальных изображений для adversarial attack для различных типов маскирований
* Исследование результатов работы модели с применением таких изображений.

# 3. Используемая модель для атак

В данной работе используется модель ResNet-50 с предобученными весами IMAGENET1K\_V2 из библиотеки PyTorch. Она имеет показатель точности top-1 порядка 80.858% на оригинальном датасете Imagenet и имеет 25.6 миллионов параметров.

# 4. Описание алгоритма для генерации изображений

В работе исследовалось изображение размером 64x64 с тремя каналами RGB. Для поиска был применен метод градиентного спуска. В качестве параметров для оптимизации использованы значения пикселей изображения в каждом из каналов. Псевдокод для процесса обучения представлен ниже.

image = random\_normal(3, 64, 64)  
for epochs in range(EPOCHS):  
 batch = []  
 labels = []  
 for class\_ind in num\_classes:  
 batch.append(mask(image, class\_ind))  
 labels.append(class\_ind)  
 predictions = model(batch)  
 loss = loss\_function(predictions, labels)  
 loss.backward()

image.clip(0, 1)

Во всех экспериментах использовались следующие гиперпараметры:

* Количество эпох – 5000
* Функция ошибки – CrossEntropyLoss
* Оптимизатор весов – SGD с learning rate = 0.1

# 5. Используемые маскирования

В работе были исследованы 3 типа маскирования изображения.

* Тип 1 - маскирование, когда все изображение делится на **k** равных частей и в зависимости от номера класса закрывается соответствующая ему часть изображения, то есть заполняется черными пикселями.
* Тип 2 - маскирование, когда все изображение делится на **k** вертикальных полос и в зависимости от номера класса закрывается соответствующая ему полоса.
* Тип 3 - маскирование, когда для каждого класса генерируется своя маска, состоящая из черных пикселей, стоящих на случайных местах.

# 6. Эксперименты

В каждом из экспериментов исследовалась зависимости времени генерации изображения и получаемая точность в зависимости от количества требуемых классов, необходимых для нахождения на изображении.

## 6.1 Маскирование с разбиением изображения на k равных частей

В данном эксперименте для каждого класса закрывается своя часть изображения. Соответственно при применении данной маски от модели требуется отнести такое изображение к этому наперед заданному классу.

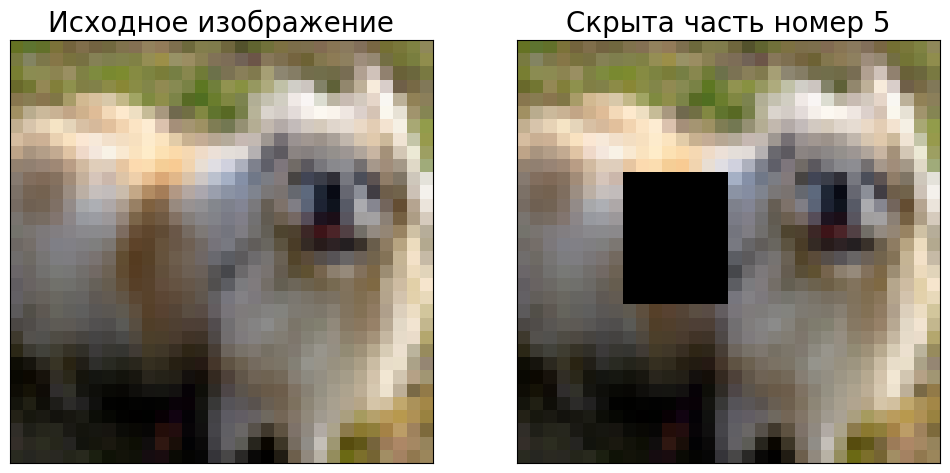


Рисунок 1. Пример маски с количеством классов равным 10. Картинка взята из датасета CIFAR-10

В эксперименте были взяты количество классов от 2 до 64 с шагом 4. Размер батча равен количеству классов. Таким образом ошибка усредняется по всем классам и градиентный спуск становится менее стохастическим.

В результате эксперимента видно, что точность нашей атаки равна почти единице при любом количестве классов, на которое разбивается маска. Это говорит о том, что искомое изображение, которое можно отнести к любому классу действительно находится. Однако количество эпох, затрачиваемое на поиск такого изображения, как правило, растет с увеличением количества классов. Флуктуации связаны со случайной инициализацией изображения.

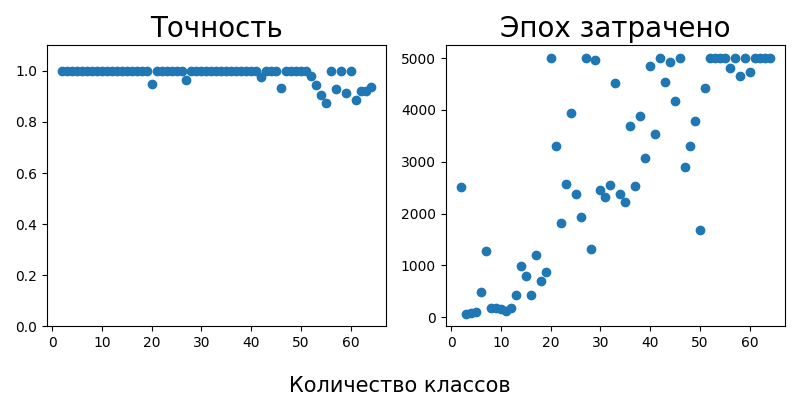


Рисунок 2. Зависимость точности и затраченных эпох при генерации изображения с маскированием типа 1

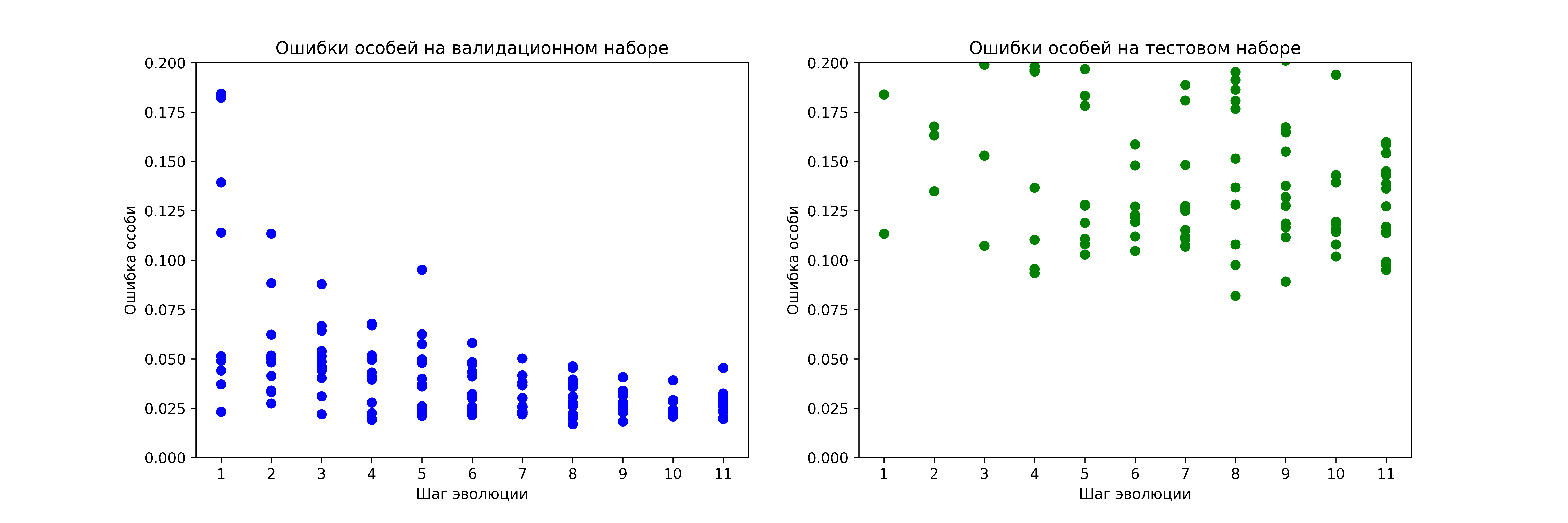
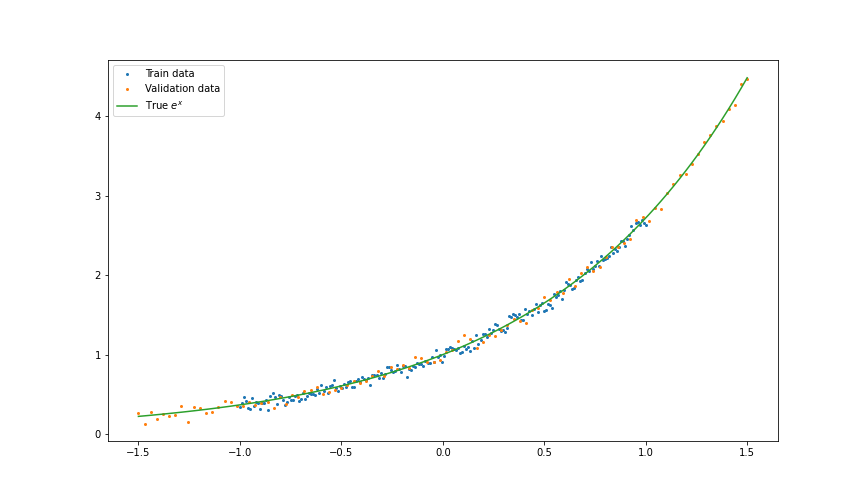
# 5. Описание реализованного алгоритма

Изначально создается популяция, генотипы особей которых генерируются случайным образов. Далее эта популяция обучается на тренировочном наборе. Все наборы данных одинаковы для любых особей любых популяций. После обучения на заданном количестве обучающих эпох происходит валидация каждой особи в популяции и определение их ошибок на этом наборе. На основе этих данных происходит эволюционный шаг — в данном случае мутация особей по ранее описанному алгоритму.

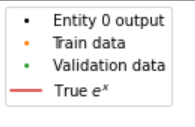
# 6. Полученные результаты

## 6.1 Задача аппроксимации

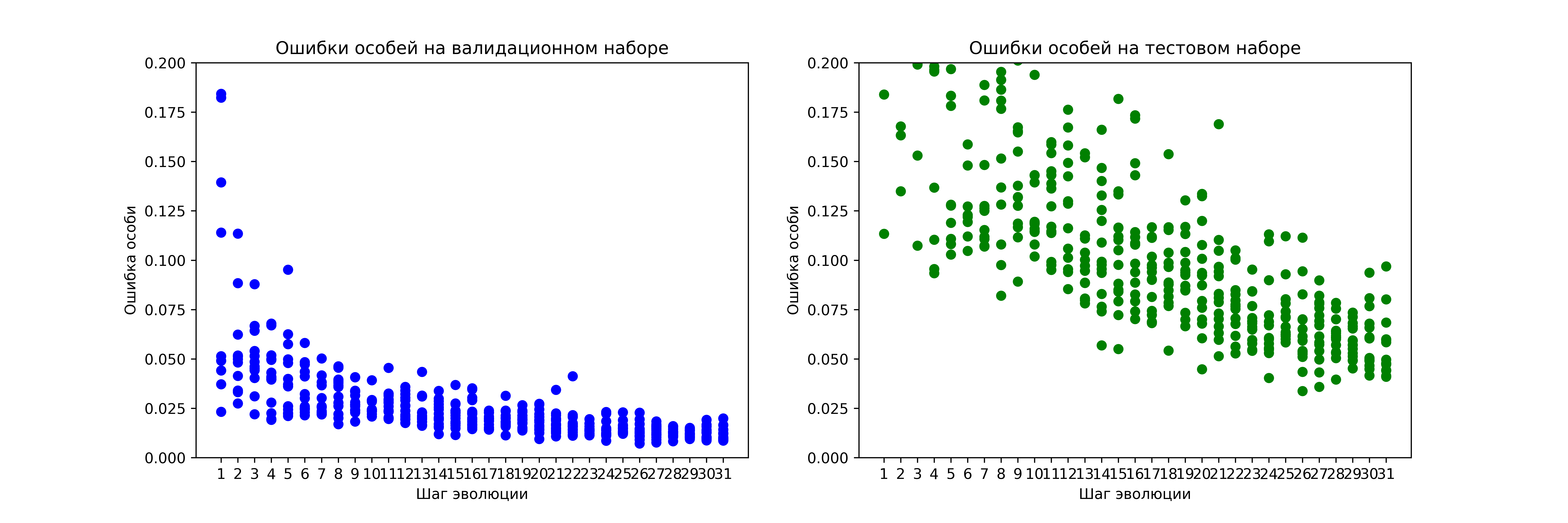
Поставим задачу аппроксимации функции.

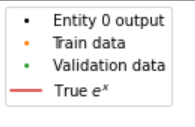
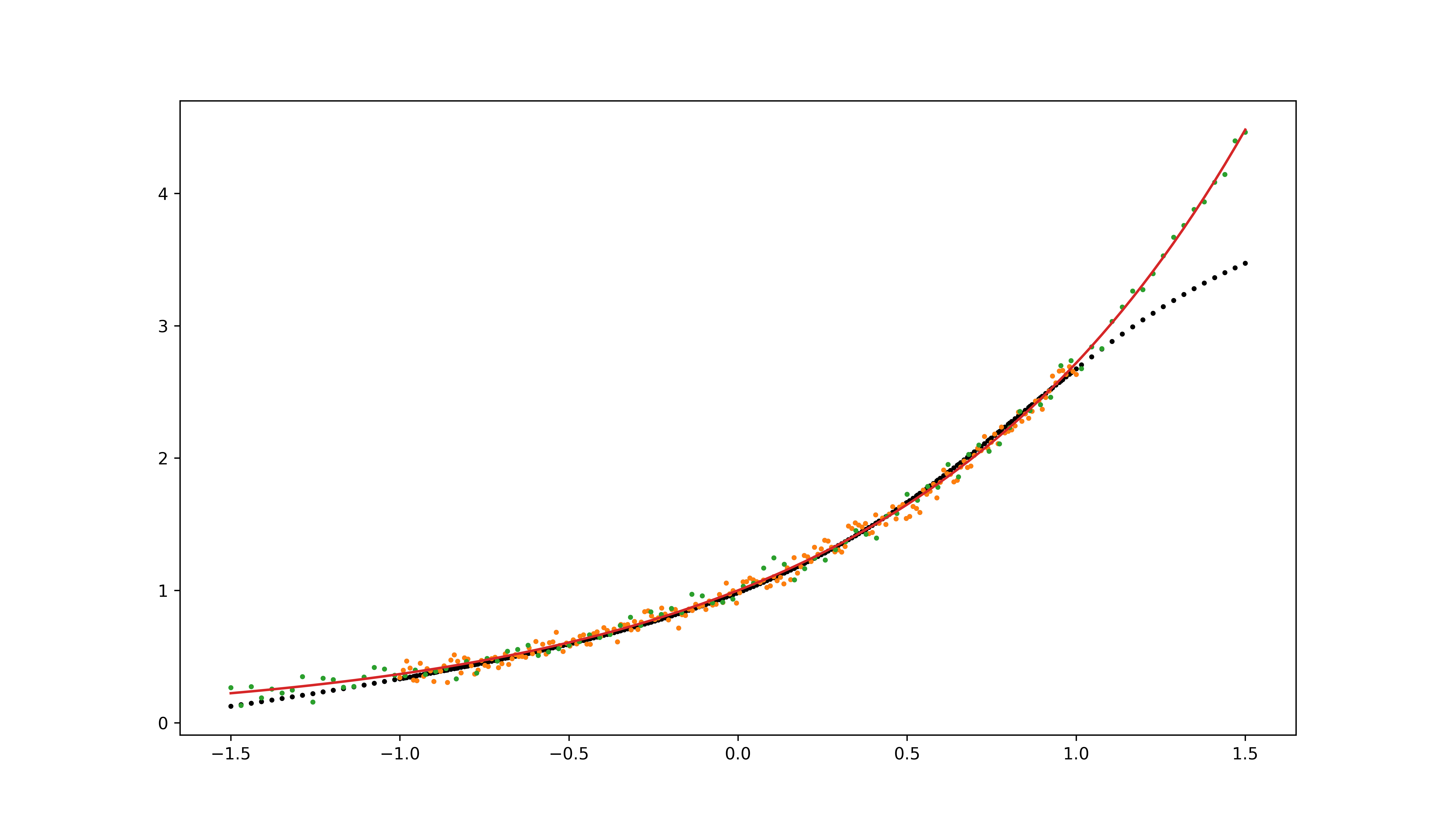
Возьмем в качестве исходной функции график с шумом.  
  
 Создадим 15 особей со случайным набором генов. Установим количество обучающих эпох равным 25 на каждую эволюционную эпоху. Спустя 11 эволюционных эпох посмотрим на результаты обучения особей.   
Как можно видеть из графика ошибки на валидационном наборе, размах ошибок среди всех особей на одной и той же эволюционной эпохе уменьшается, однако минимальная ошибка среди всех особей не сильно отличается от минимума ошибки среди всех эпох и особей. Согласно стратегии элитизма минимальная ошибка среди особей должна неувеличиваться с течением эволюционных эпох, так как особи, показавшие наилучший результат, переходят в новую популяцию без изменения архитектуры. Однако из-за того, что при переходе в новую популяцию все особи заново инициализируют, причем случайным образом, обучение идентичных особей различается, как и получавшаяся ошибка.

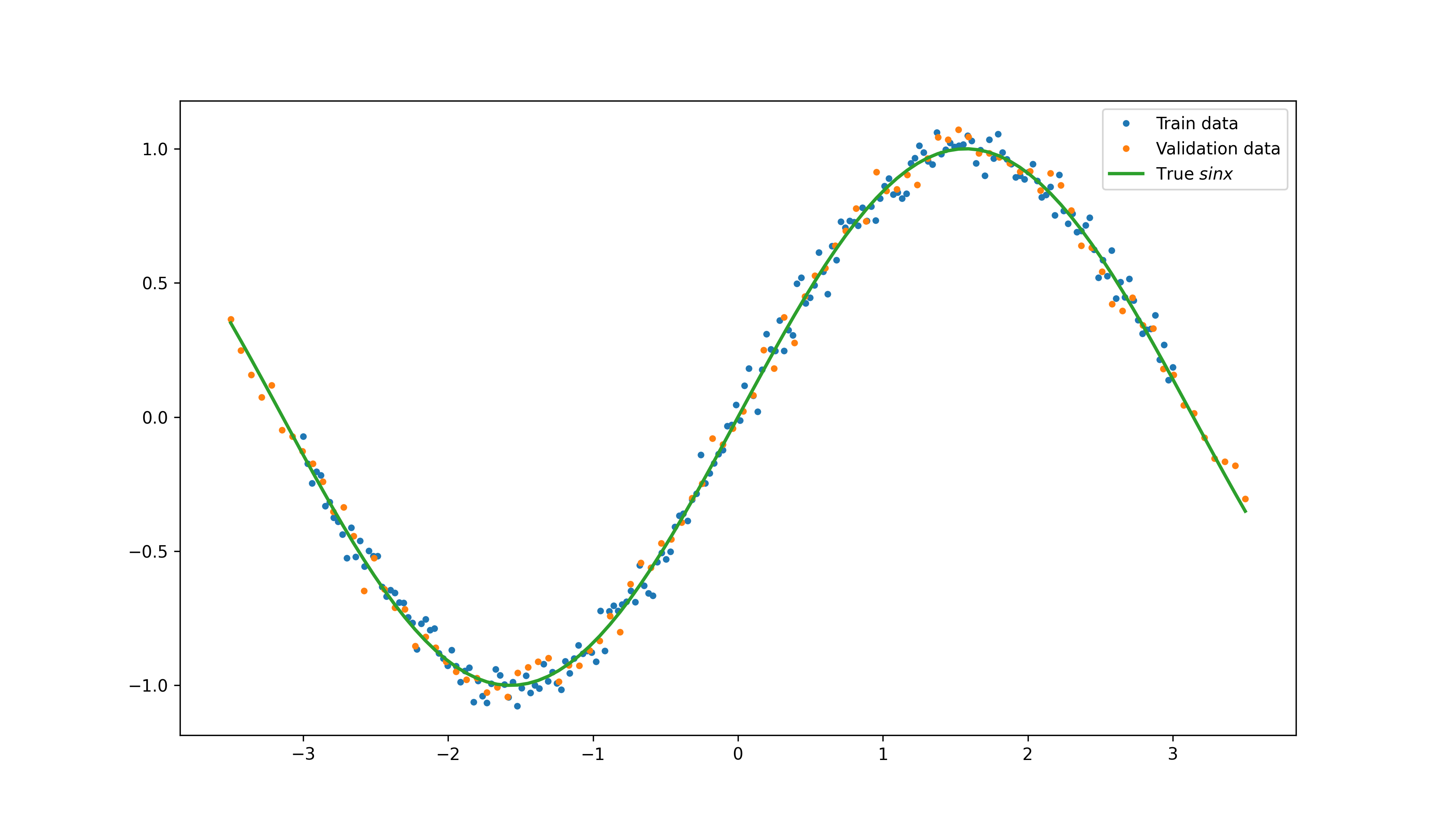
Из графика ошибки на тестовом наборе видно, что её размах внутри одной популяции также уменьшается и с количеством эволюционных эпох всё больше особей ближе к минимальной ошибке.

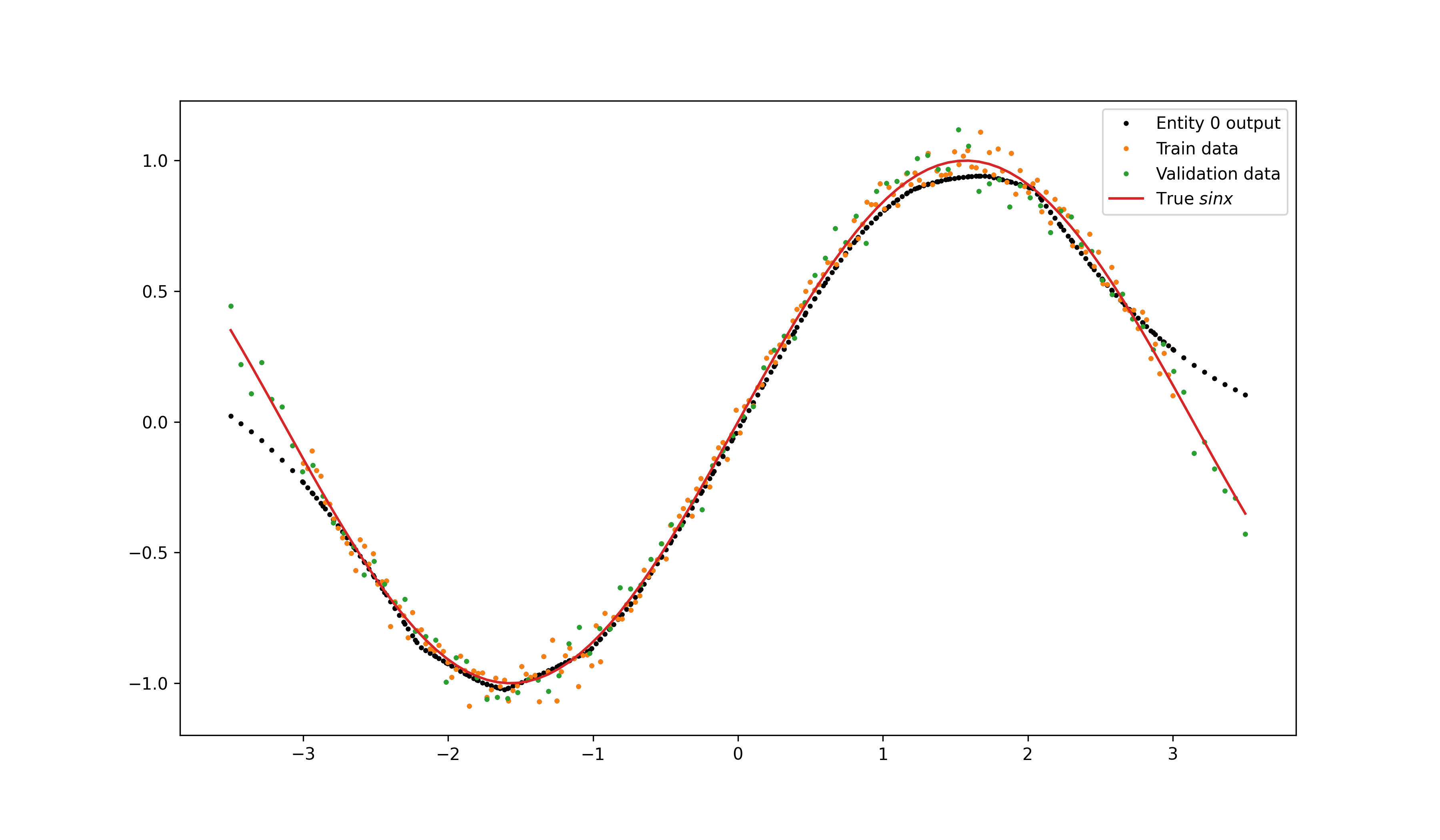
Посмотрим на результат одной из особей в последней популяции.  
 Как видно, сеть особи уже хорошо аппроксимирует исходную функцию, однако интерполяция получается неудовлетворительная.

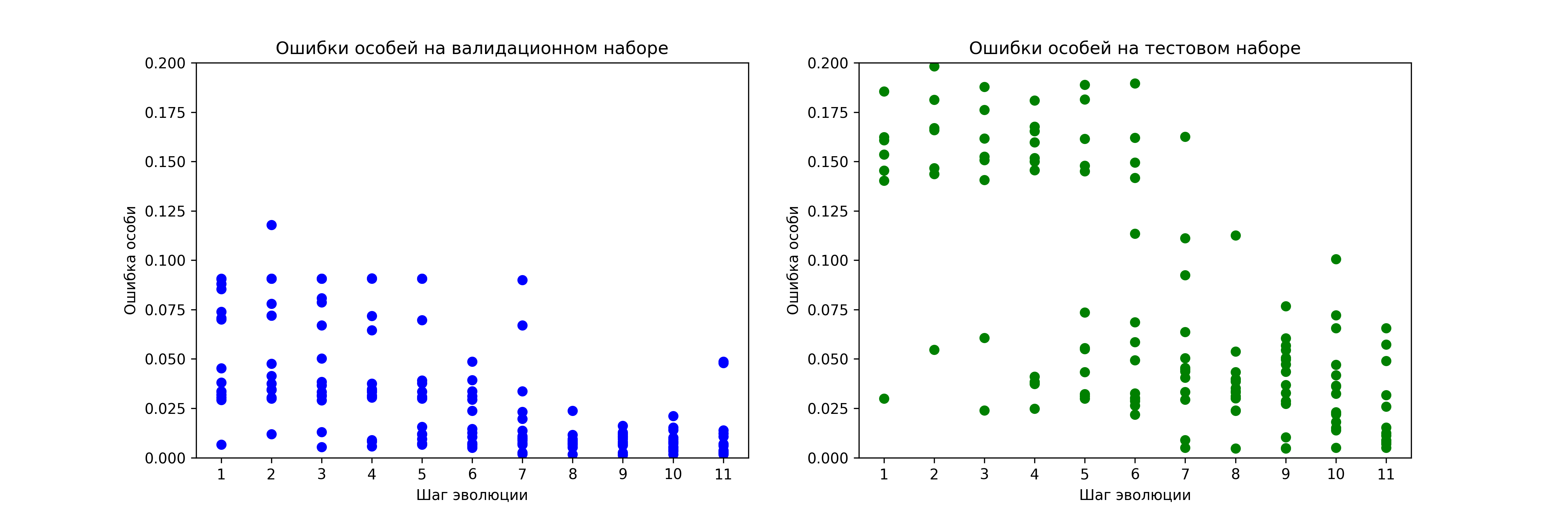
Попробуем обучить особей на ещё 20 эволюционных эпохах.

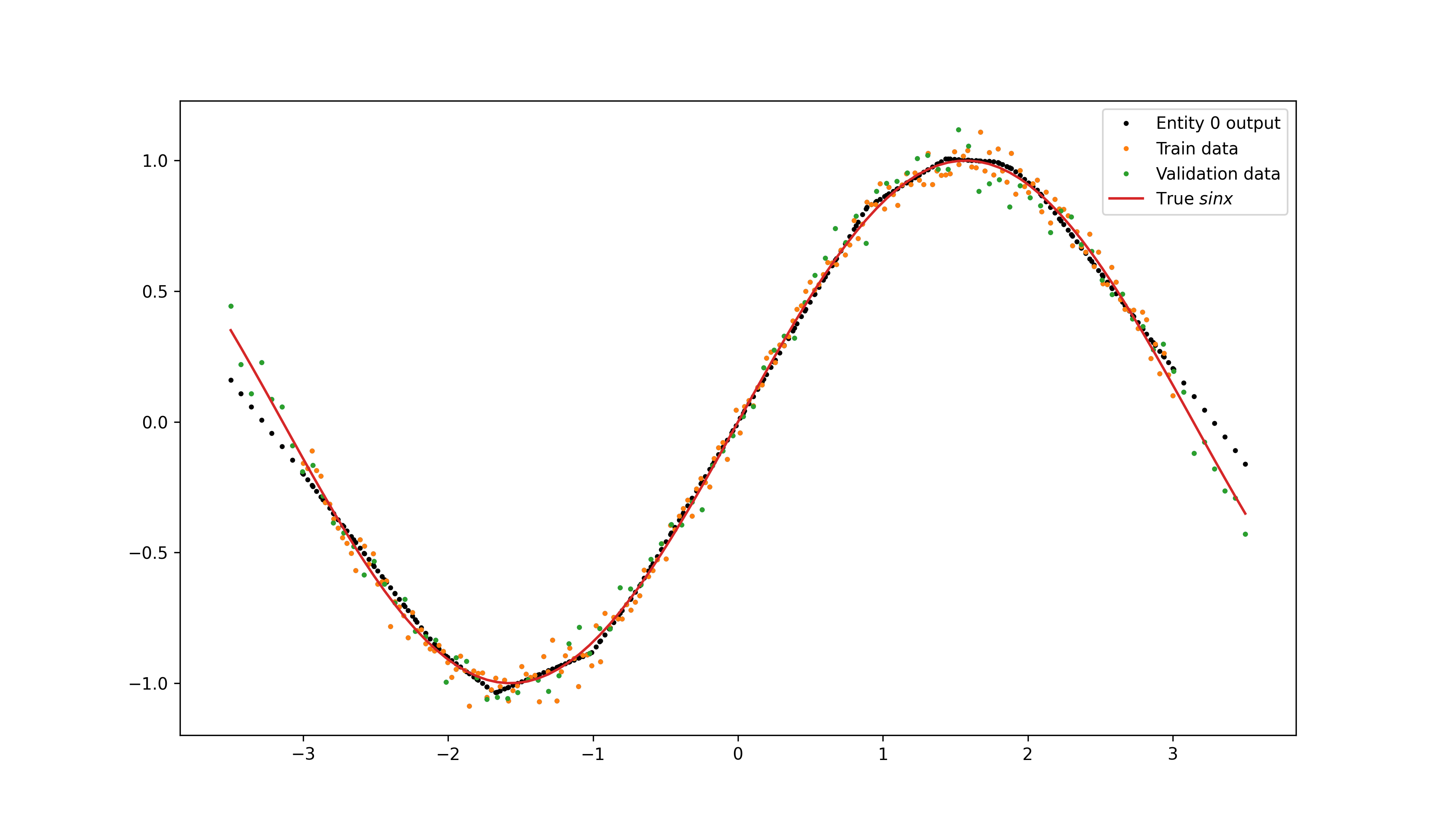
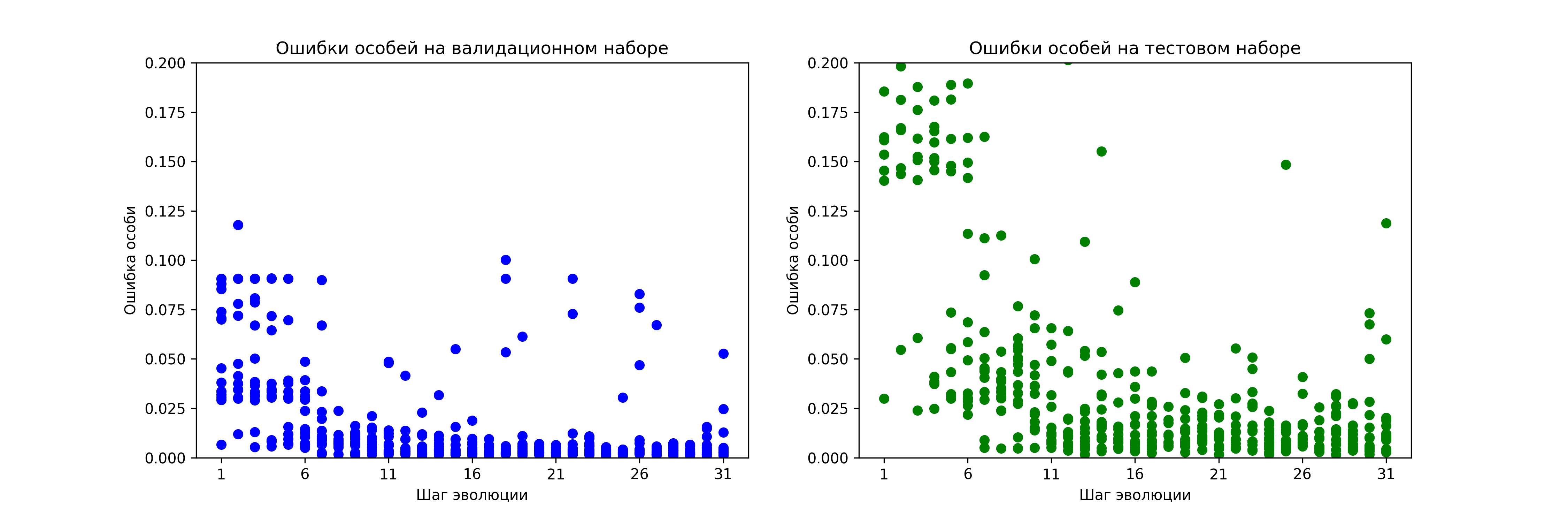
Как видно из графиков, ошибки на тестовом и валидационном наборах падают, что говорит о работоспособности генетического алгоритма на данной задаче.

Посмотрим на предсказание одной из особей на последней популяции.  
 Можно увидеть, что особь научилась очень хорошо аппроксимировать исходную функцию, даже с учетом шума, однако интерполяция все также неудовлетворительная.

Попробуем выбрать в качестве исходной функции график вида . Настройки генетического алгоритма идентичны.  
 После обучения на 11 эволюционных эпохах были получены следующие результаты.

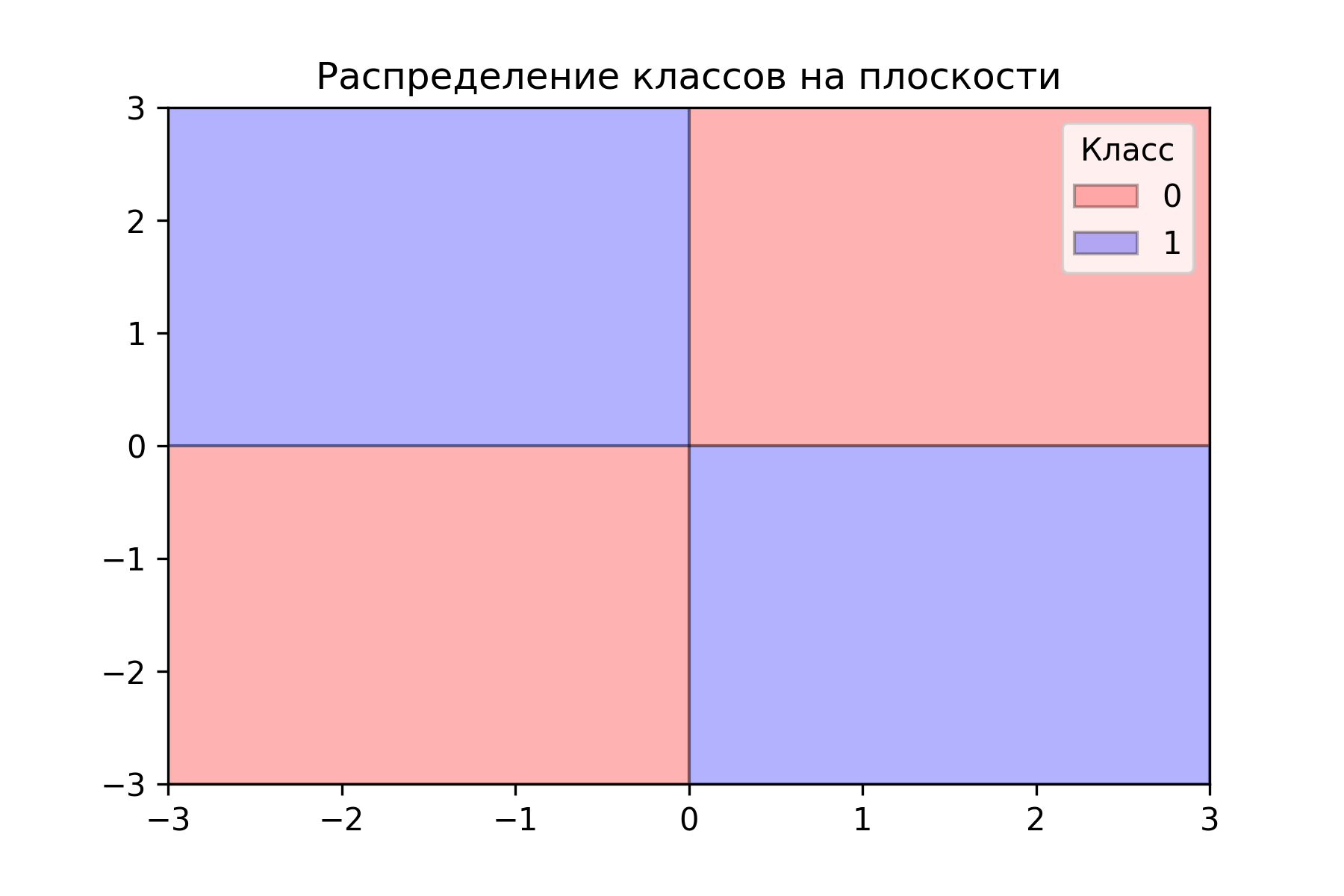
 Предсказание одной из особей из последней популяции:  
 Как можно увидеть, даже после 10 эволюционных эпох аппроксимация исходной функции достаточно хорошая.

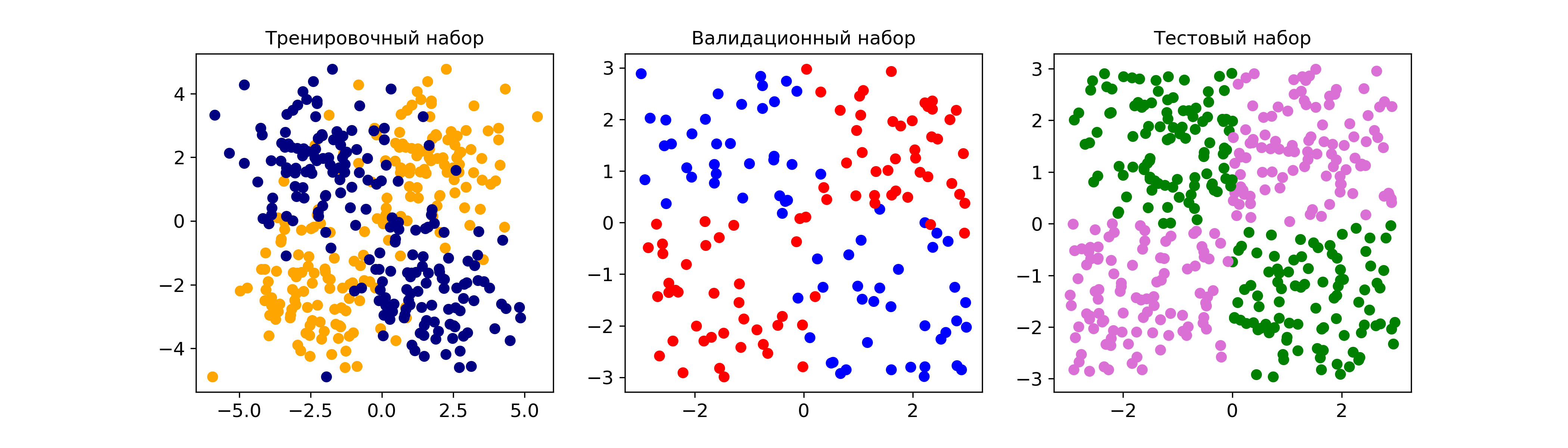
При близком рассмотрении графиков ошибок можно заметить, что размах ошибок особей внутри одной популяции уменьшается с шагом эволюции, как и при выборе экспоненты в качестве исходной функции.

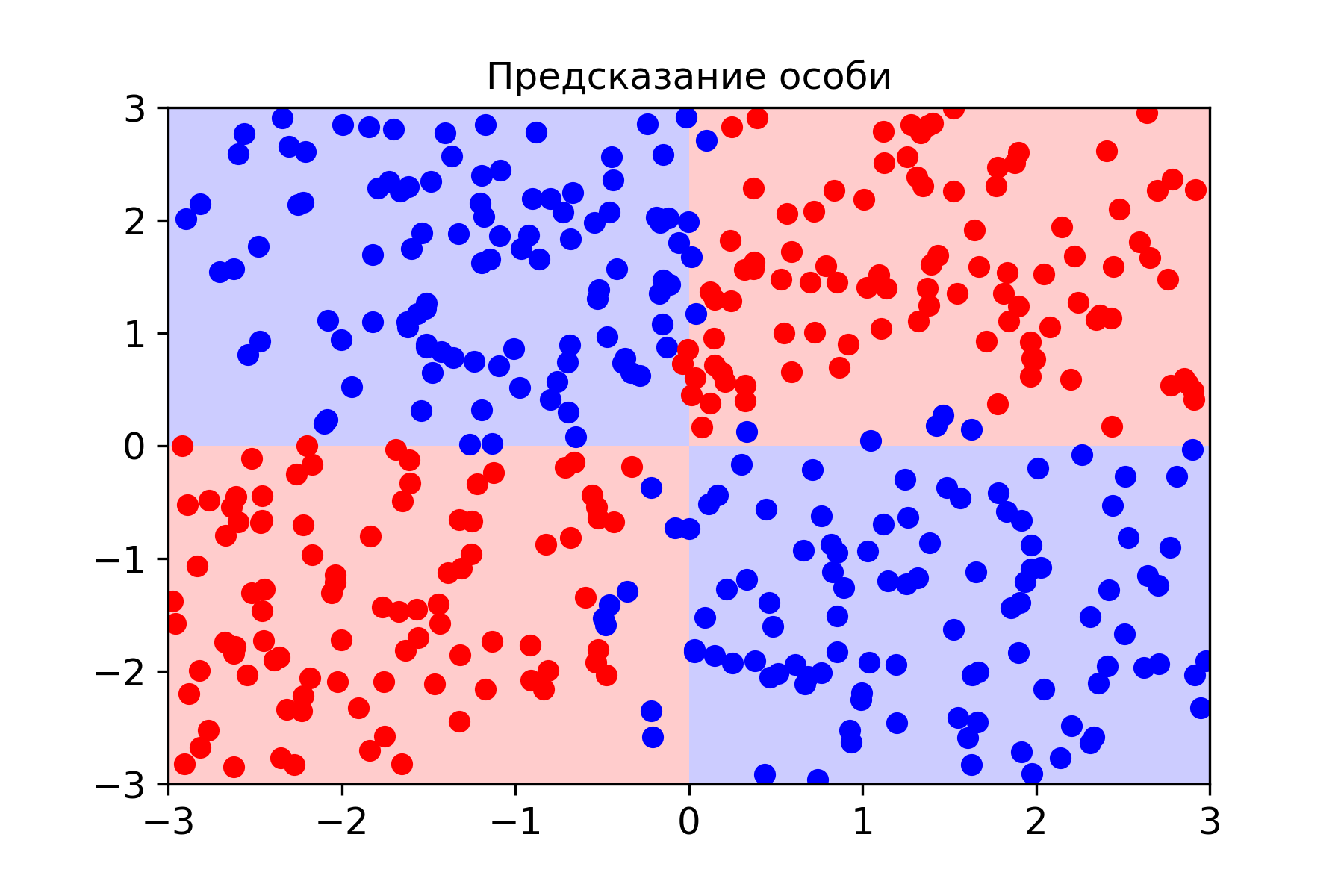
 Посмотрим на результаты спустя ещё 20 эволюционных эпох.  
 Как можно увидеть, ошибки сети уже после 10 эволюционных эпох сильно близки к 0.

Предсказание одной из особей из последней популяции:

## 6.2 Задача классификации «Исключающее ИЛИ»

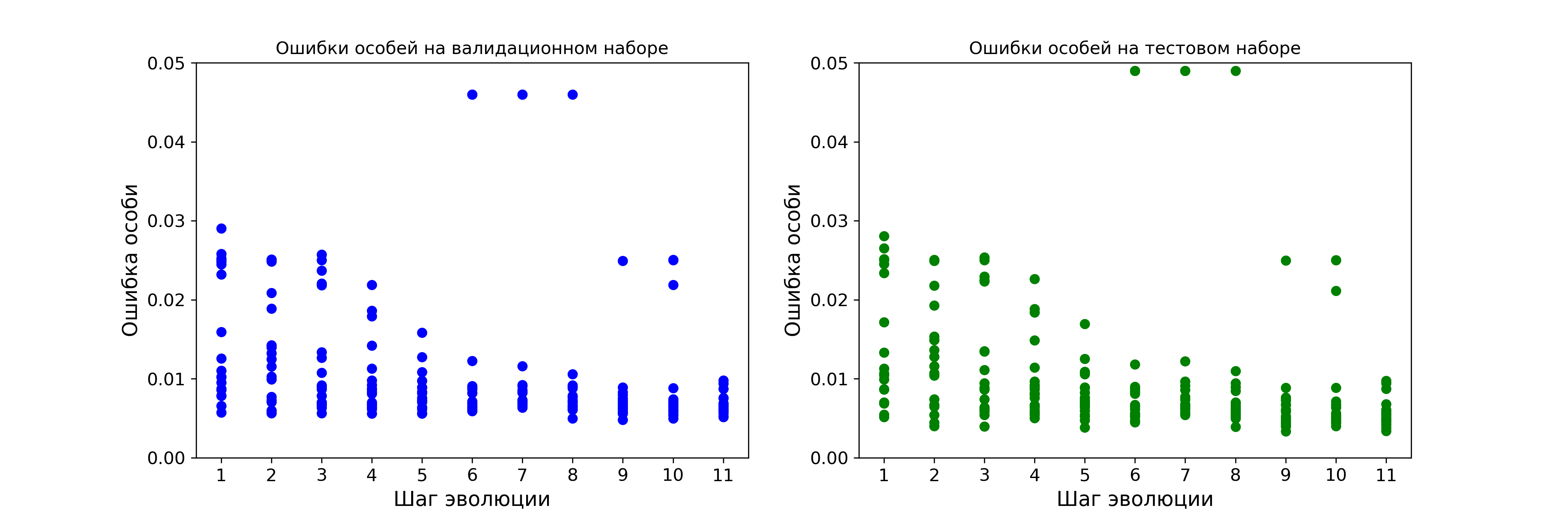
Поставим задачу классификации точек на плоскости. Распределение классов показано следующим образом:  
  
Настройки среды остаются без изменений относительно задачи аппроксимации.  
Исходя из этих областей генерируем данные для тренировочного, валидационного и тестового наборов.

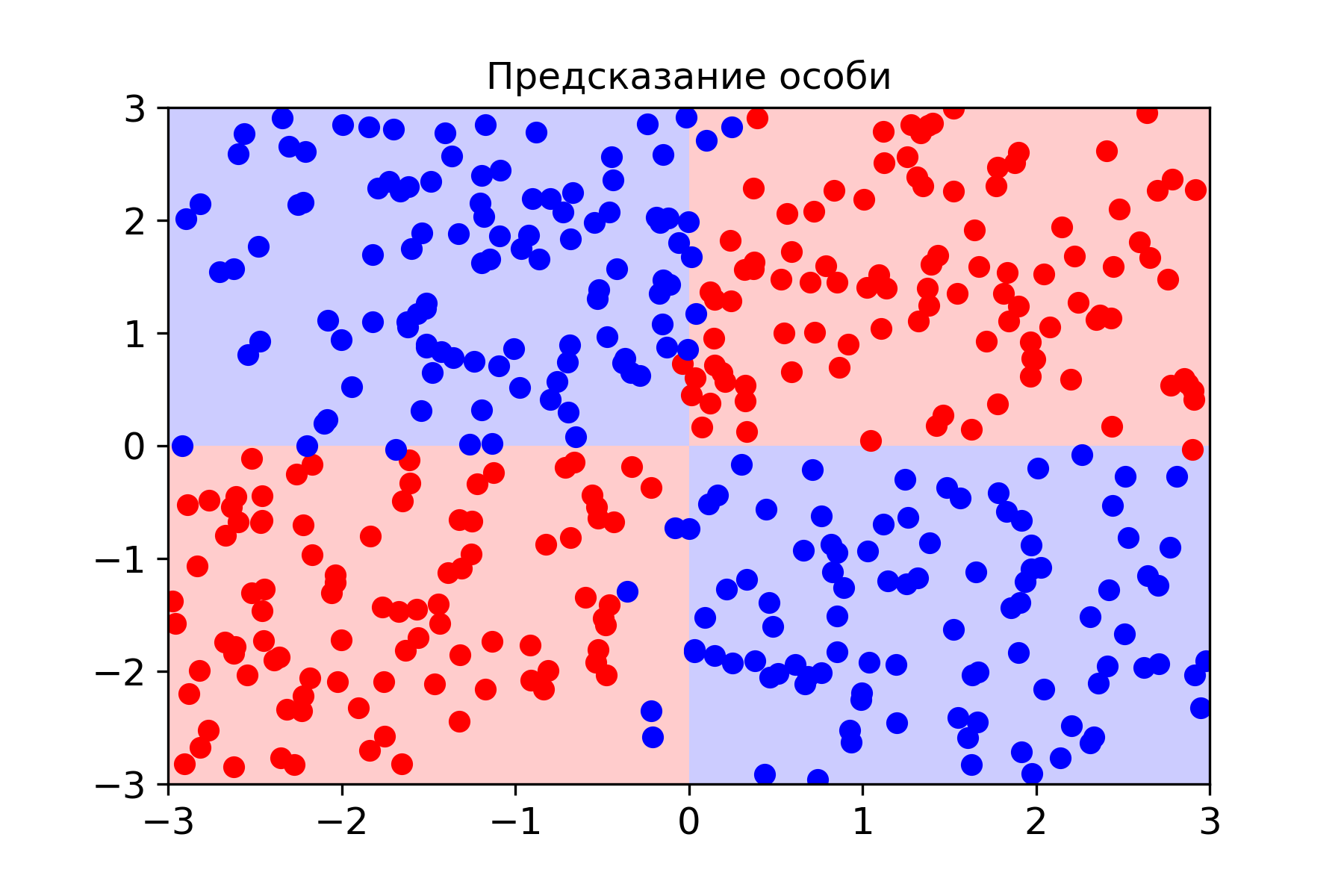
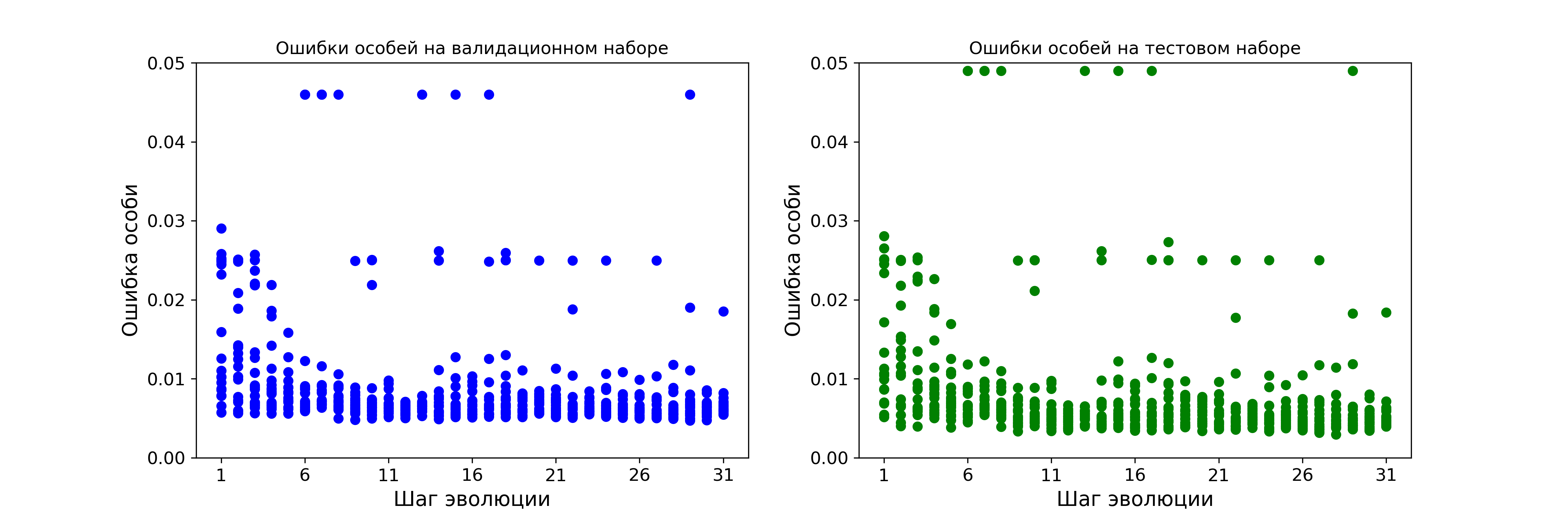
Сами наборы выглядят следующим образом:  
 Тренировочный набор сгенерирован с помощью нормального распределения с достаточно большим шумом по сравнению с валидационным набором. Тестовый набор сгенерирован без шума.

После 11 эволюционных эпох получены следующие результаты на тестовом наборе:

Цвет каждой точки обозначает принадлежность к классу.

Как можно видеть из выхода особи из последней популяции, результаты получаются достаточно адекватными.

 История ошибок особей выглядит следующим образом:  
Здесь видно, что особь, показавшая наилучший результат за все 11 эпох, появилась ещё в начале эволюции. Это говорит либо о том, что нам повезло увидеть нейронную сеть, близкую к оптимальной, уже в начале эволюции, либо о том, что методы мутации недостаточно быстро изменяют архитектуру нейронной сети.

Результаты спустя 31 эволюционную эпоху:  
  
  
  
  
  
  
  
  
  
  
  
Как видно из графика ошибок особей, видимых различий в результатах не наблюдается, что говорит о довольно устойчивой архитектуре получившихся особей.

# 7. Основные итоги работы

В результате проделанной работы была реализована модель генетического алгоритма, оптимизирующая архитектуру линейного персептрона. Задачей алгоритма было нахождение такой архитектуры НС, которая выдавала наиболее удовлетворяющее решение для поставленной задачи. Алгоритм был изучен на задачах аппроксимации функций экспоненты и синуса, а также на классификации точек на плоскости.

В условиях задачи аппроксимации функций алгоритм выделил наиболее оптимальные архитектуры НС, максимально удовлетворяющих условиям задачи за конечное время. Из графиков ошибок популяции на всех наборах можно сделать вывод, что эволюционных подход действительно находит наилучшие решения для решения задачи аппроксимации, а стратегия элитизма гарантирует сходимость данного алгоритма.

В условиях задачи классификации точек на плоскости по схеме «исключающее ИЛИ» наиболее оптимальное модель нейронной сети была получена сразу, на этапе генерации первой популяции особей. Это связано со спецификой задачи — для её решения удовлетворяет широкий класс архитектур нейронных сетей.

В заключение стоит отметить, что реализация данного алгоритма универсальна относительно задач, которые поставлены для нейронных сетей. Благодаря эволюционному подходу в переборе различных архитектур НС можно найти наиболее подходящую модель для решения поставленной задачи за конечное время. Однако стоит принять во внимание, что время генетический алгоритм основан на переборе получающихся вариантов, что в совокупности со случайной генерацией популяций даёт право утверждать, что работа алгоритма может занять продолжительное время, прежде чем будет найден наиболее оптимальное решение задачи.

# 8. Литература

1. Цой, Ю. Р. Разработка генетического алгоритма настройки искусственной нейронной сети — ВКР, Томск 2004.

2. И.Б. Бондаренко, Ю.А. Гатчин, В.Н. Гераничев, Синтез оптимальных искусственных нейронных сетей с помощью модифицированного генетического алгоритма, - УДК 519.816

3. Towardsdatascience. Genetic Algorithms + Neural Networks = Best of Both Worlds. https://towardsdatascience.com/gas-and-nns-6a41f1e8146d