

Задания заключительного этапа направления “Искусственный интеллект” Олимпиады “Я - профессионал” Категория "Бакалавриат"

1 Тернаризация нейронных сетей. 10 баллов

Условие задачи

Задача предоставлена Научно-исследовательским институтом системных исследований РАН при поддержке Российской нейросетевой ассоциации. В последнее время большое внимание уделяется тому, как сделать большие нейронные сети менее требовательными к памяти и вычислительным ресурсам. Особенно это касается больших языковых моделей, объём которых занимает гигабайты. Популярным подходом сжатия нейронных является понижение разрядности весов, в том числе бинаризация или тернаризация весов. Суть тернаризации в том, чтобы заменить веса нейронных сетей из формата float32 на числа -1, 0 или +1 (К примеру, такой же приём использовался в недавней статье «The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits»).

Задача состоит в следующем. Предположим, у нас есть некоторый слой нейронной сети, веса в котором имеют распределение Лапласа с нулевым средним: $p(w) = \frac{1}{2\lambda} e^{-\frac{|w|}{\lambda}}$, $\lambda > 0$.

Необходимо посчитать оптимальное значение порога w_0 , такое что при тернаризации:

$$w' = \begin{cases} +1, & w \geq w_0, \\ -1, & w \leq -w_0, \\ 0, & \text{else.} \end{cases}$$

корреляция между исходным весом w и финальным весом w' была максимальной.

Входные данные

В единственной строке входного файла записано действительное число $\lambda > 0$ типа данных float.

Выходные данные

Единственная строка с одним действительным числом w_0 — оптимальным значением порога.

Примечания

Ваш ответ будет засчитан как правильный, если отличается от верного ответа не более чем на 10^{-5} по абсолютному значению.

Комментарий к решению

Оптимальное значение порога вычисляется с помощью прямой оптимизации функции корреляции по параметру w_0 и равно $w_0 = \lambda$.

Пример решения на языке Python

```
a = float(input())  
  
print(a)
```

2 Интересная игра. 10 баллов

Условие задачи

Задача предоставлена Школой глубокого обучения ФПМИ МФТИ.

Петя и Вася играют в следующую игру. Каждый из них выбрал себе строчку из нулей и единиц одинаковой длины N . Далее они проводят серию бросков монетки и записывают результаты бросков в строку: при выпадении орла записывается 0, при выпадении решки — 1. Если после очередного броска монетки последние N символов, записанные в строке, совпадают со словом одного из игроков, этот игрок выигрывает.

Вам необходимо по данным строкам Пети и Васи посчитать вероятность победы Пети.

Входные данные

В единственной строке входного файла записаны две различные строки a и b длины $1 \leq N \leq 6$ из нулей и единиц, разделённые пробелом — строки Пети и Васи.

Выходные данные

Единственная строка с одним действительным числом p — вероятностью победы Пети.

Примечания

Ваш ответ будет засчитан как правильный, если отличается от верного ответа не более чем на 10^{-5} по абсолютному значению.

Комментарий к решению

Для удобства будем считать, что изначально на доске написано N символов “X”. Для строки s длины N из нулей, единиц и символов “X” (символы X должны образовывать префикс строки s некоторой длины), обозначим через $P(s)$ вероятность победы Пети при условии, что последние N символов, написанных на доске, составляют строку s . Тогда

$$P(s) = \begin{cases} (P(s[1 : N] + "0") + P(s[1 : N] + "1"))/2, & s \neq a, b; \\ 1, & s = a; \\ 0, & s = b. \end{cases}$$

Тогда можно записать систему линейных уравнений на $P(s)$ для всех возможных строк s , решение которой для $s = "XX...X"$ будет ответом на задачу.

Пример решения на языке Python

```
import numpy as np  
import os  
  
def generate_all_init_strs(n):  
    all_strs = [""]
```

```

left_bound = 0
right_bound = 1
for i in range(n):
    for char in ["0", "1"]:
        all_strs += [
            string + char
            for string in all_strs[left_bound:right_bound]
        ]
        left_bound = right_bound
        right_bound = len(all_strs)
return all_strs

```

```

def generate_row_simple(all_strs, idx, n):
    root = all_strs[idx]
    left = root + "0"
    right = root + "1"
    if len(left) > n:
        left = left[1:]
        right = right[1:]
    idx_left = all_strs.index(left)
    idx_right = all_strs.index(right)

    row = np.zeros(len(all_strs))
    row[idx] += 1
    row[idx_left] -= 0.5
    row[idx_right] -= 0.5

    return row

```

```

def generate_row_terminal(all_strs, idx, n):
    row = np.zeros(len(all_strs))
    row[idx] = 1
    return row

```

```

def generate_matrix_and_free_term(word_1, word_2, all_strs):
    n = len(word_1)

    all_rows = []
    free_terms = []
    for idx in range(len(all_strs)):
        root = all_strs[idx]
        if root == word_1:
            all_rows.append(generate_row_terminal(all_strs, idx, n))
            free_terms.append(1)
        elif root == word_2:
            all_rows.append(generate_row_terminal(all_strs, idx, n))
            free_terms.append(0)
        else:
            all_rows.append(generate_row_simple(all_strs, idx, n))
            free_terms.append(0)

```

```

    return np.stack(all_rows), np.array(free_terms)

def solve(word_1, word_2, full_output=False):
    n = len(word_1)
    all_strs = generate_all_init_strs(n)

    matrix, free_terms = generate_matrix_and_free_term(
        word_1, word_2, all_strs
    )

    probs = np.linalg.solve(matrix, free_terms)

    answer = probs[0]

    if full_output:
        return all_strs, matrix, free_terms, probs, answer
    else:
        return answer

def main():
    w_1, w_2 = input().split(" ")
    print(solve(w_1, w_2))

if __name__ == "__main__":
    main()

```

3 Бутстрепная оценка F1. 10 баллов

Условие задачи

Задача предоставлена Школой глубокого обучения ФПМИ МФТИ.

Лиза изучает возможности новой большой языковой модели ProfiGPT. Лиза хочет проверить, сможет ли модель в режиме zero-shot определить, кто написал данное сообщение — Лиза или её друг Юра.

Для этого она собрала небольшой датасет D из n размеченных сообщений d_1, \dots, d_n , подала каждое сообщение d_i в нейронную сеть с дополнительным запросом вида «Чьё это сообщение, Лизы или Юры?» и получила следующие результаты: из n примеров модель

- на a примеров правильно ответила «Лизы» (True Positive);
- на b примеров неправильно ответила «Лизы» (False Positive);
- на c примеров неправильно ответила «Юры» (False Negative);
- на $n - a - b - c$ примеров правильно ответила «Юры» (True Negative).

Получив данные числа, Лиза хотела подсчитать значение метрики $F_1 = F_1(D)$ получившегося бинарного классификатора сообщений, но дело в том, что число n довольно мало, поэтому Лиза не может быть уверена в статистической значимости получившегося результата. Для оценки возможной ошибки посчитанного значения F_1 Лиза решила воспользоваться методом бутстрэпных выборок. Этот метод основан на процедуре генерации

новой бутстрэпной выборки $E = (e_1, \dots, e_n)$ из обучающей выборки $D = (d_1, \dots, d_n)$, состоящей в следующем.

- Из выборки D равновероятно выбирается случайное сообщение d_i , положим $e_1 := d_i$.
- Аналогично и независимо друг от друга и от e_1 определяются элементы e_2, \dots, e_n .

Получившаяся выборка E скорее всего будет содержать повторяющиеся сообщения, но это не страшно. Теперь по выборке E мы можем вычислить значение $F_1(E)$.

Повторив такую процедуру N раз, мы получим N значений $X = (x_1, \dots, x_N)$ метрики F_1 на различных бутстрэпных выборках. Дисперсия выборки X называется бутстрэпной оценкой дисперсии изучаемой величины (в данном случае, F_1 -меры нашего классификатора).

Для фиксированной выборки D Лиза хочет вычислить предел значения бутстрэпной оценки дисперсии при $N \rightarrow \infty$. Помогите ей!

Входные данные

В единственной строке входного файла записаны 4 натуральных числа: общее количество сообщений n ($4 \leq n \leq 100$), количество True Positive сообщений a ($1 \leq a \leq 100$), количество False Positive сообщений b ($1 \leq b \leq 100$) и количество False Negative сообщений c ($1 \leq c \leq 100$). Гарантируется также, что $n - a - b - c > 0$.

Выходные данные

Единственная строка с одним действительным числом p — искомым пределом значения бутстрэпной оценки дисперсии с ростом N .

Примечания

Ваш ответ будет засчитан как правильный, если отличается от верного ответа не более чем на 10^{-5} по абсолютному значению.

F_1 -мера определяется как

$$F_1 = \begin{cases} \frac{2 \cdot TP}{2 \cdot TP + FP + FN}, & 2 \cdot TP + FP + FN \neq 0 \\ 0, & 2 \cdot TP + FP + FN = 0, \end{cases}$$

что совпадает с классическим определением F_1 -меры как среднего гармонического Precision и Recall, но при этом определено для любой выборки.

Комментарий к решению

Обозначим через p долю TP -примеров, а через q — долю $FP + FN$ -примеров. Пусть ξ — случайная величина, равная количеству TP -примеров, а η — случайная величина, равная количеству $FP + FN$ -примеров. Заметим, что

$$P(\xi = x, \eta = y) = C_n^x \cdot C_{n-x}^y \cdot p^x q^y (1 - p - q)^{n-x-y},$$

при $x + y \leq n$ и 0, иначе. Отсюда можно вычислить математическое ожидание и математическое ожидание квадрата случайной величины $\phi = F_1(\xi, \eta) = \frac{2\xi}{2\xi + \eta}$. Искомая дисперсия равна $E\phi^2 - (E\phi)^2$.

Пример решения на языке Python

```
import numpy as np
import os
import math

def log_choose(n, x):
    return (
        math.lgamma(n + 1) -
        math.lgamma(x + 1) -
        math.lgamma(n - x + 1)
    )

def get_prob(n, x, y, p, q):
    if x + y > n:
        return 0
    return np.exp(
        log_choose(n, x) +
        log_choose(n - x, y) +
        x * np.log(p)
        + y * np.log(q)
        + (n - x - y) * np.log(1 - p - q)
    )

def calc_f1(x, y):
    if x + y == 0:
        return 0
    else:
        return 2 * x / (2 * x + y)

def get_expectations(n, p, q):
    expect = 0
    expect_sq = 0
    for x in range(n + 1):
        for y in range(0, n - x + 1):
            expect += calc_f1(x, y) * get_prob(n, x, y, p, q)
            expect_sq += (calc_f1(x, y)**2) * get_prob(n, x, y, p, q)
    return {"expect": expect, "expect_sq": expect_sq}

def solve(n, a, b, c):
    p = a / n
    q = (b + c) / n
    expectations = get_expectations(n, p, q)
    return expectations['expect_sq'] - expectations['expect'] ** 2

def main():
    n, a, b, c = [int(item) for item in input().split()]
    print(solve(n, a, b, c))

if __name__ == "__main__":
    main()
```

4 Поиск требуемых распределений. 10 баллов

Условие задачи

Задача предоставлена Научно-образовательным центром когнитивного моделирования МФТИ.

Языковая модель представляет собой нейронную сеть, обученную на больших текстовых данных для генерации текста на естественном языке. Такая модель имеет некоторое распределение длины генерируемого текста. Пусть дана коллекция из N моделей m_1, \dots, m_N , длина выхода модели m_i имеет **дискретное равномерное распределение** Q_i на отрезке $[a_i, b_i]$.

Целевое распределение, заданное пользователем, представляет желаемое распределение длины сгенерированного текста P . Среди представленной коллекции найдите такую языковую модель, распределение длины выхода которой ближе всего к целевому **дискретному равномерному распределению** P на отрезке $[a, b]$, заданному пользователем. Близость произвольной языковой модели и целевого распределения определяется в соответствии с расстоянием Кульбака — Лейблера:

$$KL[P||Q] = \sum_{x \in \mathcal{X}} p(x) \ln \frac{p(x)}{q(x)},$$

где $p(x)$ и $q(x)$ - соответствующие функции массы вероятностей P и Q .

Таким образом, необходимо найти модель, расстояние Кульбака — Лейблера которой к целевому запросу **минимально**.

Входные данные

Первая строка содержит два целых числа $0 \leq a \leq b \leq 1000$, разделённые пробелом — интервал равномерного дискретного распределения, заданного пользователем.

Вторая строка содержит натуральное число $1 \leq N \leq 10000$ — количество языковых моделей в коллекции.

На следующих N строках вводятся через пробел два целых числа $0 \leq a_i \leq b_i \leq 1000$, разделённые пробелом — интервалы равномерных дискретных распределений, которые аппроксимируют выходы языковых моделей.

Выходные данные

Необходимо вывести одно число — ****минимальное**** расстояние Кульбака — Лейблера к запросу пользователя среди моделей в коллекции.

Примечания

Ваш ответ будет засчитан как правильный, если отличается от верного ответа не более чем на 10^{-5} по абсолютному значению. Для корректности определения расстояния Кульбака-Лейблера, гарантируется, что для всех i справедливо $a_i \leq a \leq b \leq b_i$.

Пример решения на языке Python

```
import math

class UniformDistribution:
    def __init__(self, a, b):
        self.a = a
```

```

self.b = b

def kl_uniform_discrete(dist_p, dist_q):
    """ calculate KL[P||Q] for discrete uniform distributions
    """ https://statproofbook.github.io/P/duni-kl
    return math.log(
        (dist_q.b - dist_q.a + 1) / (dist_p.b - dist_p.a + 1)
    )

"""
dist_p = UniformDistribution(*map(float, input().split(" ")))
N = int(input())
corpus_q = []
for _ in range(N):
    dist_q = UniformDistribution(*map(float, input().split(" ")))
    corpus_q.append(kl_uniform_discrete(dist_p, dist_q))

print(sorted(corpus_q)[0])

```

5 Активное обучение и фрукты. 20 баллов

Условие задачи

Задача предоставлена участником Альянса в сфере искусственного интеллекта — компанией ПАО "Сбербанк".

Вася очень любит апельсины, бананы, яблоки, мандарины и грейпфруты. В городе, где живёт Вася, целых 10050 магазинов фруктов, и везде текущие цены на фрукты разные и независимые друг от друга.

В каждом магазине города продается талончик, позволяющий в любой выходной в ближайшие 10 лет прийти в этот магазин, предъявить талончик, заплатить 100 рублей и получить 1 килограмм любого фрукта в этом магазине.

Вася написал сложную функцию $f(a_1, a_2, a_3, a_4, a_5)$, которая вычисляет справедливую стоимость этого талончика в конкретном магазине. Он считает, что все цены ведут себя как независимые броуновские движения с одинаковыми волатильностями, равными 10, и он знает только их начальные значения в конкретном магазине. То есть

$$x_i(t) = W_i(t) + x_i(0).$$

Здесь $x_i(0) = a_i$ — цена i -ого фрукта в данном магазине в начальный момент времени, $x_i(t)$ — цена i -ого фрукта в данном магазине в момент времени t . О том, каков экономический смысл функции f , написано в примечаниях.

Из всех магазинов Вася хочет сегодня купить самый выгодный талончик, но для этого ему нужно посчитать $f(a_1, a_2, a_3, a_4, a_5)$ для всех магазинов, а вычисления работают очень медленно.

Вам предстоит построить аппроксимацию функции, работающую достаточно быстро, чтобы заполнить ей тестовые значения функции. Изначально вам дано 50 обучающих точек и результаты функции, которые Вася успел посчитать, далее вы можете вызывать функцию в любых точках (в том числе и тестовых) самостоятельно.

Входные данные

Ссылка на данные.

Вам даны следующие файлы:

`train_points.csv`

Файл содержит описания 50 магазинов, для которых Вася успел вычислить значения функции f .

Первая строка файла содержит названия признаков a_1, a_2, a_3, a_4, a_5 , где a_i — цена i -ого фрукта в данном магазине в начальный момент времени. Следующие 50 строк содержат значения признаков a_i для конкретных магазинов.

`train_answers.csv`

Файл содержит единственную колонку с названием f — значения функции f на обучающих данных.

`test_points.csv`

Файл имеет структуру, аналогичную `train_points.csv` и содержит значения a_i для тестовых данных.

`baseline.ipynb`

Файл содержит реализацию функции f , а также базовое решение, которое создаёт файл `sample_submission.csv`.

Выходные данные

Вам необходимо послать в констест файл `submission.csv`, который содержит предсказанные значения f для объектов из `test_points.csv`. Файл должен иметь такой же формат, как и `train_answers.csv`. Код генерации `sample_submission.csv` содержится в файле `baseline.ipynb`.

Примечания

1. Обратите внимание, что функция выдает близкие, но не одинаковые результаты для одинаковых аргументов. В ответах, с которыми будет сравниваться ваша аппроксимация, функция посчитана 8 раз в каждой точке и усреднена.
2. Формально, справедливая стоимость талончика — это цена бермудского опциона на максимум из x_i со страйком 100 и с постоянной ставкой дисконтирования 0.1. Держатель опциона выбирает оптимальный момент, когда ему купить самый дорогой фрукт за 100 рублей (или не покупать вообще):

$$f(a_1, a_2, a_3, a_4, a_5) = \sup_{\tau \in T} \mathbb{E} \left[e^{-0.1\tau} \max(0, \max_i(x_i(\tau)) - 100) \mid \forall i : x_i(0) = a_i \right].$$

3. Ваше решение будет оцениваться по метрике RMSE, которая вычисляется по формуле

$$\sqrt{\frac{\sum (y_{true} - y_{pred})^2}{n}}.$$

Ваш итоговый балл равен:

$$\max\left(0, \frac{(\max_rmse - rmse_score) \cdot 20}{\max_rmse}\right),$$

где $\max_rmse = 25$.

После окончания констеста баллы прошкалируются в соответствии с баллом наилучшего решения.

Идея решения

Оптимальное решение достигается с помощью методов активного обучения, где модель T_i генерирует ответы на всей выборке, а затем точки, в которых уверенность модели T_i минимальная, отправляются на медленную доразметку.

6 Планирование обустройства месторождений. 20 баллов

Условие задачи

Задача предоставлена участником Альянса в сфере искусственного интеллекта — компанией ПАО "Газпром нефть".

Для добычи нефти недостаточно только лишь пробурить скважину. Не менее важной задачей является строительство сопутствующей инфраструктуры для решения множества различных задач. Но перед тем, как ее строить, необходимо распланировать проведение строительства, а, учитывая масштабы строительства, это совсем не простая задача, так как необходимо учитывать сложные взаимосвязи видов работ. Обычно планированием занимаются эксперты, а Ваша задача — помочь им находить ошибки в построенных планах.

В качестве обучающей выборки вам даны примеры **корректных** планов работ. В тестовой выборке вам необходимо выделить те планы работ, которые корректными не являются.

Входные данные

Ссылка на данные

Вам предоставляются следующие файлы:

1. `train.json`;
2. `test.json`;
3. `baseline.ipynb`.

Рекомендуем сразу открыть файл `baseline.ipynb`, потому что текст, написанный ниже, станет с ним гораздо понятнее.

`train.json`

Файл `train.json` содержит список корректных планов проведения работ: $[P_1, P_2, \dots, P_\ell]$. Каждый план P_i представляет из себя список работ по дням $[d_{i,1}, d_{i,2}, \dots, d_{i,n_i}]$, где d_{ij} — структура, которая описывает проведённые работы по i -ому плану в j -ый день.

В свою очередь, каждое d_{ij} представляет из себя `dict`, в котором ключами являются идентификаторы категорий работ, выполненных по i -ому плану в j -ый день, а значениями — конкретные виды работ, относящихся к данной категории, которые выполнялись по i -ому плану в j -ый день.

Пример одного из элементов d_{ij} :

```
{
  'J8588460': ['T6561009', 'T7892263', 'T2099010'],
  'J4088297': ['T4465784']
}
```

Данная структура означает, что в данный день по данному плану было совершено 4 работы, относящиеся к двум различным категориям. Для категорий работ идентификатор начинается с «J», для конкретной работы идентификатор начинается с «T». Например, категорией работ, соответствующей идентификатору «J8588460», может быть «прокладка нефтепровода», а конкретными работами в этой категории, соответствующими идентификаторам «T6561009», «T7892263», «T2099010» — «рытье траншей», «монтаж нагнетающего оборудования», «укладка труб» и пр.

Обратите внимание, что ID категорий работ и конкретных работ не уникальны, то есть, например, «укладка труб» (конкретная работа) будет иметь одинаковый ID для всех планов и может встречаться в разных категориях работ. Аналогично и с категориями работ: в разных планах используются одни и те же ID категорий в случае, если сами категории совпадают. При этом гарантируется, что одни и те же категории работ будут иметь одинаковый идентификатор во всех планах работ (аналогично и для конкретных работ).

Общая структура файла `train.json`:

```
[
  [
    {
      'J*****': [ 'T*****', ... ] ,
      ...
    },
    ...
  ],
  ...
]
```

test.json

Структура файла `test.json` совпадает со структурой файла `train.json`, но в некоторых планах присутствуют ошибки.

baseline.ipynb

Файл `baseline.ipynb` содержит пример загрузки данных и «наивное» решение задачи.

Выходные данные

Вам необходимо для каждого плана предсказать следующие характеристики:

- Для каждого **дня** необходимо предсказать, содержит ли он хотя бы одну ошибку;
- Для каждого уникального идентификатора **категории** работ необходимо предсказать, содержала ли данная категория работ хотя бы одну ошибку в хотя бы один из дней.

Вам необходимо сформировать файл `submission.json`. Файл должен быть представлен в виде списка, где каждый элемент соответствует плану из тестовой выборки. Элемент списка должен содержать `dict`, в котором должны быть ключи “`is_correct_day`” и “`is_correct_job`”. Значение, соответствующее ключу “`is_correct_day`”, должно содержать список значений типа `bool`, причем длина списка должна равняться количеству дней в соответствующем плане. Значение, соответствующее ключу “`is_correct_job`” должно содержать словарь, в котором ключами являются все **категории** работ плана, а значениями — элементы типа `bool`, обозначающие корректность этой категории работ внутри всего плана.

Файл `baseline.ipynb` содержит пример формирования файла необходимого формата.

Примечания

Оценка за решение задачи выставляется следующим образом:

$$10 \cdot \frac{F_1(\text{jobs_pred}, \text{jobs_true}) - \text{baseline}_1}{1 - \text{baseline}_1} + 10 \cdot \frac{F_1(\text{days_pred}, \text{days_true}) - \text{baseline}_2}{1 - \text{baseline}_2},$$

где

- *jobs_pred* — одномерный массив предсказанных вами корректностей работ по всем планам;
- *jobs_true* — массив истинных корректностей работ;
- *days_pred* — одномерный массив предсказанных вами корректностей всех планов в конкретные дни;
- *days_true* — массив истинных корректностей планов по дням;
- *baseline₁* = 0.857 — значение F_1 -меры по работам у `sample_submission.json`;
- *baseline₂* = 0.934 — значение F_1 -меры по дням у `sample_submission.json`.

После окончания контеста баллы прошкалируются в соответствии с баллом наилучшего решения.

Идея решения

Для решения задачи следует использовать методы детекции выбросов на основе признаков, созданных на основе взаимосвязей между различными работами в одной категории.

7 Потенциальное поле. 20 баллов

Условие задачи

Задача предоставлена Институтом искусственного интеллекта AIRI.

Такие алгоритмы планирования как Model Predictive Control (MPC) могут обеспечить локальное планирование движения мобильных роботизированных платформ. Для их функционирования в каждой точке пространства необходимо знать аналитическое представление потенциала, который представляет из себя стоимость столкновения робота с этой точкой для случая когда и карта препятствий, и след робота произвольны. Следом робота называют область, занимаемую им на карте.

В данной задаче предлагается построить модель нейронной сети, которая возвращает дифференцируемую стоимость столкновения в каждой точке пространства в зависимости от позы робота, карты препятствий и формы робота.

Входные данные

Ссылка на данные [Альтернативная ссылка на данные](#)

В качестве данных для обучения предлагается набор из 198 карт и 2083 точек с позицией агента и со значениями потенциалов для каждой карты.

В файле `train_maps.npy` находятся карты размера (50, 50). Размерность 11 - это фреймы движения динамического препятствия на карте. В условиях данной задачи его можно воспринимать как статическое препятствие.

В файле `train_data.npy` находятся координаты точек (X,Y,Theta) для которых известен потенциал, который необходимо предсказать. Размерность 10 - это различные вариации ориентации робота в данной точке.

В файле `train_answers.npy` находятся значения потенциалов для всех точек массива `train_data.npy`

ВНИМАНИЕ! Данные довольно тяжёлые. Рекомендуем поставить их скачиваться заранее.

Выходные данные

Вам нужно выполнить предсказание потенциала всех точек из файла `test_data.npy`. Соответствующие этим точкам карты находятся в файле `test_maps.npy`.

Примечания

Метрикой качества решения является Mean Squared Error.

Ваша посылка оценивается по формуле

$$20 \cdot \frac{\text{baselinescore} - \text{score}}{\text{baselinescore}},$$

где *score* — значение MSE, которого достигает ваше решение, а *baselinescore* = 0.035.

После окончания конкурса баллы прошкалируются в соответствии с наилучшим решением.

Идея решения

Оптимальное решение достигается с помощью двухбашенной нейронной сети, которая кодирует картинку лабиринта свёрточной нейронной сетью и положение робота полностью связными слоями.