Задания заключительного этапа направления “Искусственный интеллект” Олимпиады “Я - профессионал” Категория "Магистратура/специалитет"

1 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 $1 \leqslant N \leqslant 10$ 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

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

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()
```

2 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 $d_1, \ldots, d_n$, 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 $E = (e_1, \ldots, e_n)$ из обучающей выборки $D = (d_1, \ldots, d_n)$, 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 $e_2, \ldots, e_n$.

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 $X = (x_1, \ldots, x_N)$ 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 $N \to \infty$. Помогите ей!

Входные данные

В единственной строке входного файла записаны 4 натуральных числа: общее количество сообщений n ($4 \leqslant n \leqslant 100$), количество True Positive сообщений a ($1 \leqslant a \leqslant 100$), количество False Positive сообщений b ($1 \leqslant b \leqslant 100$) и количество False Negative сообщений c ($1 \leqslant c \leqslant 100$). 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

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

при $x + y \leqslant n$ 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()
```

3 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 нормальное распределение с нулевым средним и дисперсией σ^2:

$$p(w) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{w^2}{2\sigma^2}}.$$

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

$$w' = \begin{cases} +1, & w \geqslant w_0, \\ -1, & w \leqslant -w_0, \\ 0, & else. \end{cases}$$

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

Входные данные

В единственной строке входного файла записано действительное число σ в десятичной записи с точкой в качестве десятичного разделителя: суммарное количество символов в строке не превосходит 10.

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^{-4} по абсолютному значению.

Комментарий к решению

Ясно, что оптимальное значение порога пропорционально σ и равно $C\sigma$. Константа C вычисляется оптимизацией некоторой функции с помощью кода

```
from scipy.optimize import minimize

def f_correlation(x):
    t = x / (2**0.5 * sigma)
    return 2 * math.exp(-t**2) / math.sqrt(2 * math.pi * (1 - math.erf(t)))

xx = np.linspace(0, 2, 100)
rho = [f_correlation(x) for x in xx]
```

```
solution = minimize(fun=lambda x: -f_correlation(x), x0=0.5, tol=1e-12)
print(solution['x'][0]
```

Пример решения на языке Python

```
a = float(input())

print(a * 0.6120031)
```

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 $m_1, \ldots, m_N$, длина выхода модели m_i после нормировки имеет *нормальное распределение* $Q = \mathcal{N}(\mu_i, \sigma_i^2)$.

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 нормальному распределению $P = \mathcal{N}(\mu, \sigma^2)$, заданному пользователем. Близость произвольной языковой модели и целевого распределения определяется в соответствии с расстоянием Кульбака — Лейблера:

$$KL[P||Q] = \int_{\chi} p(x) \ln \frac{p(x)}{q(x)} dx,$$

где $p(x)$ и $q(x)$ — соответствующие функции плотности вероятностей P и Q.

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

Входные данные

Первая строка содержит два вещественных числа типа `float` через пробел — параметры нормального распределения μ и σ^2, заданные пользователем.

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

На следующих N строках вводятся через пробел два вещественных числа μ_i и σ_i^2 — параметры нормальных распределений, которые аппроксимируют языковые модели $m_1, \ldots, m_N$.

Выходные данные

Необходимо вывести одно действительное 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.

Пример решения на языке Python

```
import math


class NormalDistribution:
  def __init__(self, mean, variance):
    self.mean = mean
    self.variance = variance

def kl_normal(dist_p, dist_q):
  # calculate KL[P||Q] for normal distributions
  # https://statproofbook.github.io/P/norm-kl
  first_term = ((dist_q.mean - dist_p.mean) ** 2) / dist_q.variance
  second_term = dist_p.variance / dist_q.variance
  third_term = math.log(dist_p.variance / dist_q.variance)
  return 0.5 * (first_term + second_term - third_term - 1)


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

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

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:

$$\max(0, \frac{(max\_rmse - rmse\_score) \cdot 20}{max\_rmse}),$$

где $max\_rmse = 25$.

После окончания контеста 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

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: $[P_1, P_2, \ldots, P_\ell]$. Каждый план P_i представляет из себя список работ по дням $[d_{i,1}, d_{i,2}, \ldots, d_{i,n_i}]$, 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 ровняться 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`, обозначащие корректность этой категории работ внутри всего плана.

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

Примечания

Оценка за решение задачи выставляется следующим образом:

$$10 \cdot \frac{F_1(jobs\_pred, jobs\_true) - baseline_1}{1 - baseline_1} + 10 \cdot \frac{F_1(days\_pred, days\_true) - baseline_2}{1 - baseline_2},$$

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;
- $baseline_1 = 0.857$ — значение F_1-меры по работам у `sample_submission.json`;
- $baseline_2 = 0.934$ — 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 Полезность стратегии. 20 баллов

Условие задачи

Задача предоставлена Институтом искусственного интеллекта AIRI.

Вася занимается исследованиями в области обучения с подкреплением. В процессе его исследований он столкнулся с интересной стратегией, которую ему захотелось оценить. К сожалению у Василия нет прямого доступа к ней, т.к. он лишь видел, процесс её использования. К счастью ему удалось запомнить состояния, в нескольких коротких

траекториях при применении этой стратегии, к сожалению, без указания конкретных действий и полученных вознаграждений. А также ему удалось получить всю информацию об используемом Марковском процессе принятия решения (МППР).

Так как Василий знает автора данной стратегии, он уверен в том, что она является оптимальной для данного МППР. Однако, ему неизвестно, какой коэффициент дисконтирования использовал автор при обучении. В своих расчетах он решил опираться на стандартное значение 0.95. На этом силы исследователя закончились, поэтому ваша задача довести до конца эту непростую задачу. Вам нужно посчитать полезность каждого состояния данного МППР для рассматриваемой стратегии с заданным коэффициентом дисконтирования.

Входные данные

Ссылка на данные

В файле `mdp.json` находится рассматриваемый МППР, кодируемый двумя ключами `transition_probs` и `reward_function`. Также в данном файле находятся доступные траектории, ключ `trajectories`. В файле `submit.json` дан пример решения с полезностями, равными 0. В файле `example.py` находится пример чтения входных данных и генерации ответа.

Выходные данные

Измените файл `submit.json`, рассчитав полезность данной стратегии при коэффициенте дисконтирования, равном 0.95.

Примечания

Метрика для оценки решения участника — MSE.

Оценка решений участников осуществляется по формуле

$$score = 20 \times \left(1 - \frac{\max(w, \min(b, s)) - b}{w - b}\right).$$

Здесь: s – результат участника по метрике MSE, b – результат лучшего решения жюри по метрике MSE, $w = 0.5$ – нижний порог оценки.

Идея решения

Будем подбирать коэффициент дискотнирования γ, используя тернарный поиск.

При проверке γ используем итерацию по полезностям, чтобы посчитать полезности состояния действия $Q(s, a)$. По полезностям восстанавливаем стратегию. Для каждой траектории считаем с помощью стратегии и МППР вероятность того, что данная траектория могла быть получена данной стратегией. Сумма по вероятностям это значение, которое максимизируем тернарным поиском.

Полученный коэффициент дисконтирования используем, чтобы восстановить стратегию и посчитать полезности состояний МППР.