Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждения высшего образования

«Национальный исследовательский технологический университет «МИСиС»

Лабораторная работа №1.3

по дисциплине

«Основы теории информации»

на тему:

«Основная задача кодирования»

Направление подготовки:

09.03.01 «Информатика и вычислительная техника»

Семестр 5

Выполнил: Проверил:

Попов Егор Куприянов Вячеслав

Борисович \_\_\_\_ Васильевич\_\_\_\_\_\_\_\_

(Ф.И.О. студента) (Ф.И.О преподавателя)

БИВТ-22-СП-4 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(№ группы) (оценка)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата сдачи) (дата проверки)

Подпись: \_\_\_\_\_\_\_\_\_\_ Подпись: \_\_\_\_\_\_\_\_\_\_

Москва – 2022

**Цель работы:**

Ознакомится с темой «Понятие и свойства энтропии» и решить задачу №1.3 с использованием языка программирования Python.

1. **Теоретический материал:**

Информация – сведения, являющиеся объектом хранения, передачи, преобразования. Мера информации зависит от априорных вероятностей событий. Неопределенность выбора элементов 𝑥𝑖 при создании сообщения удобно характеризовать энтропией. Для опыта 𝛼 рассмотрим таблицу распределения вероятностей:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Исходы опыта | 𝑥1 | 𝑥2 | 𝑥3 | … | 𝑥𝑛 |
| Вероятности | 𝑝(𝑥1) | 𝑝(𝑥2) | 𝑝(𝑥3) | … | 𝑝(𝑥𝑛) |

Мера неопределенности равна

𝐻(𝑥) = −∑𝑃( 𝑛 𝑖=1 𝑥𝑖) ∗ 𝑙𝑜𝑔2(𝑃(𝑥𝑖))

Энтропия – мера неопределенности ситуации 𝑥𝑖 с конечным или счетным числом исходов, причем 𝑃(𝑥𝑖) – вероятность исхода x, а суммирование ведется по всем исходам.

Свойства энтропии:

1. Энтропия опыта равна нулю лишь в том случае, когда одна из вероятностей равна 1, а все остальные равны 0.
2. Энтропия опыта максимальна, если вероятности исходов отвечают равенству 𝑃(𝑥𝑖 ) = 1 𝑛 , то есть они равновероятны.
3. Энтропия непрерывна и всегда положительна.

Энтропия совместного события 𝑥 и 𝑦

𝐻(𝑥, 𝑦) = − ∑𝑃( 𝑖,𝑗 𝑥𝑖 , 𝑦𝑗) ∗ 𝑙𝑜𝑔2(𝑃(𝑥𝑖 , 𝑦𝑗))

Условная энтропия – энтропия, определяемая при условии, что становятся известны исходы другого события y, усредненная по всем значениям x.

𝐻𝑦 (𝑥) = − ∑𝑃(𝑥𝑖 , 𝑦𝑗) ∗ 𝑙𝑜𝑔2(𝑃𝑥𝑖 (𝑦𝑗)) 𝑖,𝑗

Количество информации

𝐼 = 𝐻(𝑥) − 𝐻апост(𝑥)

Где 𝐻(𝑥) – энтропия до получения эксперимента;

𝐻апост(𝑥) = 𝐻(𝑥, 𝑦) – энтропия после проведения эксперимента.

При отсутствии помех при передаче сообщения

𝐻апост(𝑥) = 0 и 𝐼 = 𝐻(𝑥).

Другими словами, количество информации – это мера уменьшения неопределенности события 𝑥, возникающего вследствие того, что становятся известными исходы другого события 𝑦, усреднение по исходам 𝑥 и 𝑦.

1. **Условие задачи:**

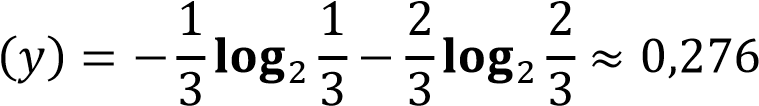
Пусть опыт *y* состоит в извлечении одного шара из урны, содержащей 5 чёрных и 10 белых шаров, опыт 𝑥𝑘 – в предварительном извлечении из той же урны (без возвращения обратно) *k* шаров. Чему равна энтропия опыта y и количество информации об этом опыте, содержащейся в опытах 𝑥1, 𝑥2,

𝑥13, 𝑥14?

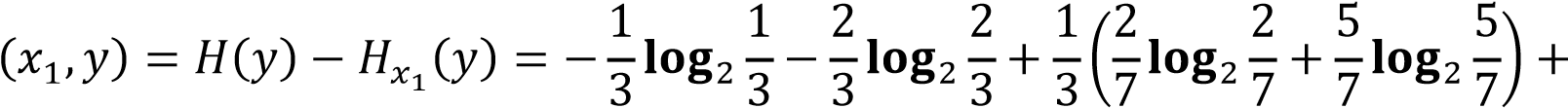
1. **Практическая часть:**

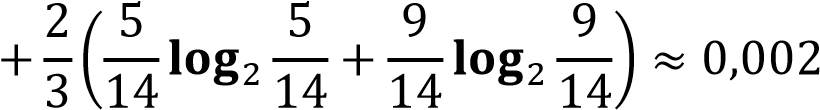
**3.1 Математическое решение**

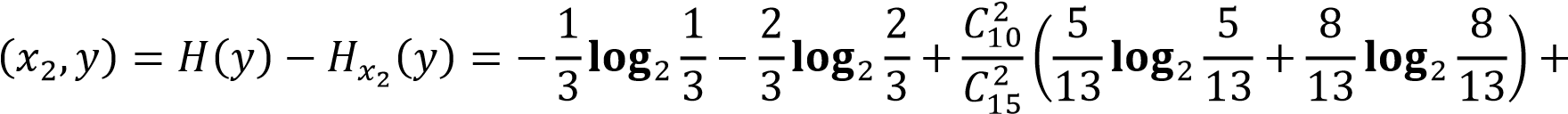
Энтропия опыта *y* очевидно равна

𝐻.

Далее

𝐼

;

𝐼

1 𝐶1 4 2 3

+ 𝐶2 (13𝐥𝐨𝐠2 13 + 13 𝐥𝐨𝐠2 13) + 𝐶152 (13 𝐥𝐨𝐠2 13 + 13 𝐥𝐨𝐠2 13) 0,0025;

𝐶

10

5

4

9

9

𝐶

5

3

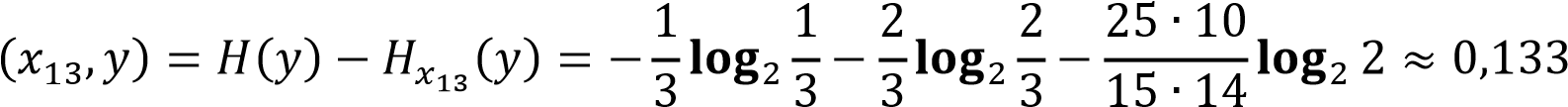
10

10

≈

15

где 𝐶𝑛𝑘 – число групп по *k* шаров, которые можно выбрать из общего числа *n* шаров.

𝐼;

𝐼(𝑥14, 𝑦) = 𝐻(𝑦) − 𝐻𝑥14(𝑦)𝐻(𝑦) ≈ 0,276;

**3.2 Программная реализация** Программная реализация содержит 3 файла:

* Entropy
* ConditionalEntropy
* Main

Импортируемые библиотеки:

* Math – для функции логарифмирования
* Itertools – для работы с сочетаниями

|  |
| --- |
| import math  import itertools as it |

|  |
| --- |
| class Entropy:      def \_\_init\_\_(self, balls\_numbers):          self.balls = balls\_numbers          self.total\_colors = len(balls\_numbers)          self.number\_all = sum(balls\_numbers)      def probability(self, value):          return value / self.number\_all      def log2(self, value):          if not value:              return 0          return math.log10(value)      def entropy(self):          res = 0          for i in range(self.total\_colors):              res -= (self.probability(self.balls[i]) \* self.log2(self.probability(self.balls[i])))          return res |

Функции класса:

* \_\_init\_\_(self, balls\_numbers) – конструктор класса, принимает массив с количествами шаров разных цветов
* probability(self, value) – возвращает вероятность события для данного набора шаров, принимает кол-во шаров
* log2(self, value) – возвращает логарифм по вероятности, принимает кол-во шаров • entropy(self) – возвращает энтропию для данного набора шаров Поля класса:
* balls – массив с количествами шаров
* total\_colors – кол-во цветов шаров
* number\_all – кол-во шаров

**3.2.2 ConditionalEntropy**

|  |
| --- |
| class ConditionalEntropy:      def \_\_init\_\_(self, balls\_numbers, k=0):          self.k = k          self.balls = balls\_numbers          self.number\_all = sum(balls\_numbers)      def form\_begin(self):          if self.k > self.balls[0]:              return self.k - self.balls[0]          else:              return 0      def form\_end(self):          if self.k <= self.balls[1]:              return self.k          else:              return self.balls[1]      def form\_k(self):          if self.k <= self.balls[0]:              return self.k          elif self.balls[0] < self.k <= self.balls[1]:              return self.balls[0]          else:              return self.number\_all - self.k      def c\_index(self, a, b):          C5 = len(list(it.combinations([''] \* self.balls[0], a)))          C10 = len(list(it.combinations([''] \* self.balls[1], b)))          C15 = len(list(it.combinations([''] \* self.number\_all, self.k)))          return C5 \* C10 / C15      def conditional\_entropy(self):          begin = self.form\_begin()          end = self.form\_end()          res = 0          for i in range(begin, end + 1):              a = self.k - i              elem1 = self.balls[0] - a              elem2 = self.balls[1] - i              temp\_entr\_arr = Entropy([elem1, elem2])              res -= self.c\_index(a, i) \* temp\_entr\_arr.entropy() \* -1          return res |

Функции класса:

* \_\_init\_\_(self, balls\_numbers, k=0) – конструктор класса, принимает массив с количествами шаров разных цветов и кол-во изьятых шаров
* form\_begin(self) – формирует начальное значение для формирования сочетаний
* form\_end(self) – формирует конечное значение для формирования сочетаний
* form\_k(self) – формирует индекс для формирования сочетания
* c\_index(self, a, b) – рассчитывает сочетания для выборки шаров
* conditional\_entropy(self) - возвращает условную энтропию для данного набора шаров Поля класса:
* k – кол-во шаров, изъятых из урны
* balls – массив с количествами шаров
* number\_all – кол-во шаров

**3.2.3 Main**

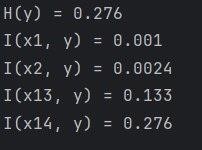
|  |
| --- |
| balls\_nums = [5, 10]  entr\_arr1 = Entropy(balls\_nums)  Hy = entr\_arr1.entropy()  print('H(y) =', round(Hy, 3))  entr\_arr\_x1 = ConditionalEntropy(balls\_nums, k=1)  entr\_arr\_x2 = ConditionalEntropy(balls\_nums, k=2)  entr\_arr\_x13 = ConditionalEntropy(balls\_nums, k=13)  entr\_arr\_x14 = ConditionalEntropy(balls\_nums, k=14)  Hx1 = entr\_arr\_x1.conditional\_entropy()  Hx2 = entr\_arr\_x2.conditional\_entropy()  Hx13 = entr\_arr\_x13.conditional\_entropy()  Hx14 = entr\_arr\_x14.conditional\_entropy()  Ix1y = Hy - Hx1  Ix2y = Hy - Hx2  Ix13y = Hy - Hx13  Ix14y = Hy - Hx14  print('I(x1, y) =', round(Ix1y, 3))  print('I(x2, y) =', round(Ix2y, 4))  print('I(x13, y) =', round(Ix13y, 3))  print('I(x14, y) =', round(Ix14y, 3)) |

Созданные объекты:

* entr\_arr1 – объект энтропии для *H(y)*
* Hy – значение энтропии для *H(y*)
* entr\_arr\_x1, entr\_arr\_x2, entr\_arr\_x13, entr\_arr\_x14 – объекты условной энтропии для *H(y)* при значениях *k* = 1, 2, 13, 14 соответственно
* Hx1, Hx2, Hx13, Hx14 – значения условной энтропии для *H(y)* при значениях *k* = 1, 2, 13, 14 соответственно
* Ix1, Ix2, Ix13, Ix14 – количество информации для опыта *y* при *k* = 1, 2, 13, 14

соответственно

Результат выполнения:



**Вывод:**

Ознакомились с темой «Понятие и свойства энтропии» и решили задачу №1.3 с использованием языка программирования Python.