Линейная алгебра. Лабораторная работа 1, осень 2020

Алгебраические структуры на компьютере на примере задачи теории кодирования

Ильдар Сафин

Часть 1. Библиотеки

В ней набираются работы для знакомства со библиотекой

- `numpy` — основная библиотека для работы с массивами
- `matplotlib` — в ней нужен `matplotlib.pyplot`, следующий неочевидный формат имений
- `sympy` — для символьных вычислений
- `IPython.display` — для красивого вывода

Подключить их можно следующим образом:

```
In [1]: %matplotlib inline
        import numpy as np
        import matplotlib.pyplot as plt
        import sympy
        from IPython.display import display, Image
        sympy.init_printing()
```

Обратите внимание, что numpy импортируется как `numpy`, а `matplotlib.pyplot` — как `plt`. Это стандартные имена, принятые среди пользователей библиотек.

Перед тем, как начать работать с библиотеками, нужно выполнить магическую команду `%matplotlib inline`, а потом — `init_printing()`. Это нужно для того, чтобы библиотеки могли выводить свои объекты в удобном виде.

Теперь можно перейти к работе с библиотеками. Для начала — немного о том, как устроены данные в библиотеке numpy.

Основной объект, с которым оперирует библиотека numpy — это `ndarray`, многомерный массив. Его можно получить, например, из списка или списка списков:

```
In [2]: a = np.array([1, 2, 3])
        a
```

```
Out[2]: array([1, 2, 3])
```

```
In [3]: b = np.array([[1, 2, 3], [4, 5, 6]])
        b
```

```
Out[3]: array([[1, 2, 3],
               [4, 5, 6]])
```

Массивы поддерживают, что называется, векторные операции:

```
In [4]: a + 1
        a * 2
        a * a
        b * a
```

```
Out[4]: array([2, 3, 4])

Out[4]: array([2, 4, 6])

Out[4]: array([1, 4, 9])

Out[4]: array([[ 1,  4,  9],
               [ 4, 10, 18]])
```

Также можно к элементам massiva `a` обращаться, как к элементам списка:

```
In [5]: a[0]
        a[1] = 5
        a
```

```
Out[5]: 1

Out[5]: array([1, 5, 3])
```

Или к элементам матрицы `b`:

```
In [6]: b[0, 1]
        b[1, 2] = 0
        b
```

```
Out[6]: 2

Out[6]: array([[1, 2, 3],
               [4, 5, 0]])
```

Или к подмассивам:

```
In [7]: b[:, 1]
        b[0, :]
        b[:, :]
```

```
Out[7]: array([2, 5])

Out[7]: array([1, 2, 3])

Out[7]: array([[1, 2, 3],
               [4, 5, 0]])
```

Обратите внимание, что массивы изменяются на месте.

Есть много других способов создания массивов. Например, `arange` аналогичен `range`, но создаёт массив:

```
In [8]: np.arange(0, 1, 0.1)
```

```
Out[8]: array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])
```

Или `linspace`, который создаёт массив из `50` элементов, равномерно покрывающих отрезок `[0, 1]`:

```
In [9]: np.linspace(0, 1, 50)
```

```
Out[9]: array([0.  , 0.02, 0.04, 0.06, 0.08, 0.1 , 0.12, 0.14, 0.16, 0.18,
               0.2 , 0.22, 0.24, 0.26, 0.28, 0.3 , 0.32, 0.34, 0.36, 0.38,
               0.4 , 0.42, 0.44, 0.46, 0.48, 0.5 , 0.52, 0.54, 0.56, 0.58,
               0.6 , 0.62, 0.64, 0.66, 0.68, 0.7 , 0.72, 0.74, 0.76, 0.78,
               0.8 , 0.82, 0.84, 0.86, 0.88, 0.9 , 0.92, 0.94, 0.96, 0.98,
               1.  ])
```

Обратите внимание, что numpy автоматически не строит.

Есть много других способов создания массивов. Например, `arange` аналогичен `range`, но создаёт массив:

```
In [10]: np.zeros(5)
         np.ones(5)
         np.eye(5)
```

```
Out[10]: array([0., 0., 0., 0., 0.])

Out[10]: array([1., 1., 1., 1., 1.])

Out[10]: array([[1., 0., 0., 0., 0.],
                [0., 1., 0., 0., 0.],
                [0., 0., 1., 0., 0.],
                [0., 0., 0., 1., 0.],
                [0., 0., 0., 0., 1.]])
```

Обратите внимание, что numpy автоматически не строит.

Есть также функции для генерации случайных векторов. Например, `random.uniform` генерирует вектор заданной длины из случайных чисел, равномерно распределённых на отрезке `[0, 1]`:

```
In [11]: np.random.uniform(0, 1, 10)
```

```
Out[11]: array([0.71518937, 0.60754485, 0.54688113, 0.76353751, 0.44797312,
                0.67159499, 0.10526531, 0.65632959, 0.57404756, 0.29414484])
```

Или `random.normal`, который генерирует вектор из `10` элементов, распределённых по нормальному закону с нулевым матожиданием и единичной дисперсией:

```
In [12]: np.random.normal(0, 1, 10)
```

```
Out[12]: array([-0.81090518, -0.30788499, -0.10622851, -1.07475137,  0.75544992,
                0.33354759,  0.31179588, -0.17337266, -0.19562389,  0.12811363])
```

Numpy также поддерживает случайные перестановки. Например, `random.shuffle` перемешивает вектор на месте:

```
In [13]: a = np.arange(10)
         np.random.shuffle(a)
         a
```

```
Out[13]: array([7, 1, 5, 6, 9, 8, 3, 4, 2, 0])
```

Numpy также поддерживает случайные выборки. Например, `random.choice` делает выборку из заданного вектора `a` размера `size` с вероятностями, заданными вектором `p`:

```
In [14]: np.random.choice(a, size=3, p=np.ones(10)/10)
```

```
Out[14]: array([1, 7, 4])
```

Или `random.permutation`, который делает случайную перестановку вектора:

```
In [15]: np.random.permutation(a)
```

```
Out[15]: array([7, 1, 5, 6, 9, 8, 3, 4, 2, 0])
```

Numpy также поддерживает агрегирующие операции. Например, `sum` суммирует все элементы массива:

```
In [16]: a.sum()
```

```
Out[16]: 45
```

Или `prod` перемножает все элементы массива:

```
In [17]: np.arange(1, 11).prod()
```

```
Out[17]: 3628800
```

Numpy также поддерживает логические операции. Например, `all` проверяет, все ли элементы массива истинны:

```
In [18]: (a > 0).all()
```

```
Out[18]: False
```

Или `any`, который проверяет, истинен ли хотя бы один из элементов массива:

```
In [19]: (a > 0).any()
```

```
Out[19]: True
```

Numpy также поддерживает индексацию массивами. Например, можно выбрать элементы `a` с номерами `0`, `2` и `4` так:

```
In [20]: a[np.array([0, 2, 4])]
```

```
Out[20]: array([7, 5, 9])
```

Или так, если нужно выбрать элементы, которые больше нуля:

```
In [21]: a[a > 0]
```

```
Out[21]: array([7, 1, 5, 6, 9, 8, 3, 4, 2])
```

Numpy также поддерживает изменение размерности массивов. Например, `reshape` меняет размерность массива, сохраняя количество элементов. Например, можно из `10` элементов сделать матрицу размерности `2 × 5`:

```
In [22]: a.reshape(2, 5)
```

```
Out[22]: array([[7, 1, 5, 6, 9],
                [8, 3, 4, 2, 0]])
```

Или `ravel`, который делает из многомерного массива одномерный:

```
In [23]: b.ravel()
```

```
Out[23]: array([1, 2, 3, 4, 5, 0])
```

Numpy также поддерживает транспонирование матриц:

```
In [24]: b.T
```

```
Out[24]: array([[1, 4],
                [2, 5],
                [3, 0]])
```

Numpy также поддерживает матричное умножение. Например, `dot` перемножает матрицы:

```
In [25]: b.dot(b.T)
```

```
Out[25]: array([[14, 19,  3],
                [19, 41, 12],
                [ 3, 12, 16]])
```

Numpy также поддерживает сортировки. Например, `sort` сортирует вектор на месте:

```
In [26]: a.sort()
         a
```

```
Out[26]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Numpy также поддерживает обращение индексов. Например, `argsort` выдаёт такой вектор, что если применить к нему индексацию, то получится отсортированный вектор:

```
In [27]: a[np.array([9, 7, 2, 1, 0])]
```

```
Out[27]: array([9, 7, 2, 1, 0])
```

Numpy также поддерживает чтение и запись файлов. Например, `save` записывает вектор в бинарный файл `filename.npy`, а `load` — читает:

```
In [28]: np.save('filename', a)
         np.load('filename.npy')
```

```
Out[28]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Numpy также поддерживает линейную алгебру. Например, `linalg.solve` решает СЛУ с матрицей `a` и свободными членами `b`:

```
In [29]: a = np.random.uniform(0, 1, (10, 10))
         b = np.random.uniform(0, 1, 10)
         np.linalg.solve(a, b)
```

```
Out[29]: array([ 1.17313149,  0.14731285, -0.69468539,  1.69314091,  0.17304552,
                0.60773343, -0.06305835,  0.29189724, -0.37127929,  0.71195482])
```

Numpy также поддерживает генерацию случайных чисел из различных распределений. Например, `random.binomial` генерирует `10` чисел, каждое из которых — это число орлов из `100` бросков монетки, выпадающей на орла с вероятностью `0.5`:

```
In [30]: np.random.binomial(100, 0.5, 10)
```

```
Out[30]: array([54, 47, 49, 56, 52, 45, 51, 49, 56, 46])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [31]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[31]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [32]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[32]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [33]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[33]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [34]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[34]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [35]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[35]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [36]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[36]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [37]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[37]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [38]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[38]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [39]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[39]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [40]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[40]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [41]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[41]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [42]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[42]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [43]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[43]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [44]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[44]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [45]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[45]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [46]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[46]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [47]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[47]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [48]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[48]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [49]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[49]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [50]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[50]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [51]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[51]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [52]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[52]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [53]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[53]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [54]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[54]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [55]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[55]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [56]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[56]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [57]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[57]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [58]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[58]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [59]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[59]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [60]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[60]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [61]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[61]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [62]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[62]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [63]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[63]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [64]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[64]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [65]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[65]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [66]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[66]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [67]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[67]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [68]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[68]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [69]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[69]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [70]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[70]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [71]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[71]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [72]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[72]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [73]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[73]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [74]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[74]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [75]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[75]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [76]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[76]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [77]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[77]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [78]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[78]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [79]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[79]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущим способом, на `10` интервалов и отдаёт частоты попадания в каждый из интервалов:

```
In [80]: a = np.random.binomial(100, 0.5, 10000)
         np.histogram(a, bins=np.linspace(0, 100, 10))
```

```
Out[80]: (array[0, 0, 0, 0, 0, 0, 0, 0, 0],
          array[0.  , 11.11, 22.22, 33.33, 44.44, 55.56, 66.67, 77.78, 88.89,
                100.  ])
```

Numpy также поддерживает вычисление гистограмм. Например, `histogram` разбивает `10000` чисел, сгенерированных предыдущ