Конспект по теме "Матрицы и матричные операции"

Создание матриц

Матрица — это прямоугольная числовая таблица, или двумерный массив. Состоит из m строк и n столбцов (размер записывается так: $m \times n$). Матрицы обычно обозначаются заглавными латинскими буквами, например, *A*. Их элементы — строчными с двойным индексом: *aij*, где *i* — номер строки, а *j* — номер столбца.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \end{pmatrix}$$

Создадим в *NumPy* матрицу из списка списков, для этого вызовем *np.array()*. Все вложенные списки одинаковой длины.

```
import numpy as np

matrix = np.array([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]])
print(matrix)
```

Вместо списка списков возьмём список векторов:

```
import numpy as np

string0 = np.array([1,2,3])
string1 = np.array([-1,-2,-3])
list_of_vectors = [string0, string1]
matrix_from_vectors = np.array(list_of_vectors)

print(matrix_from_vectors)
```

Создадим матрицу из таблицы *Pandas*: её атрибут `values` — это матрица.

```
import pandas as pd
import numpy as np

matrix = df.values
print(matrix)
```

Атрибутом `shape` определим размер матрицы A. Её элемент aij задаётся в *NumPy* как *A*[i,j]: строки и столбцы нумеруются с нуля, как и индексы массива.

```
import numpy as np

A = np.array([
    [1, 2, 3],
    [2, 3, 4]])

print('Размер:', A.shape)
print('A[1, 2]:', A[1, 2])
```

Из матрицы выделим отдельные строки и столбцы:

```
import numpy as np

matrix = np.array([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9],
    [10,11,12]])

print('Строка 0:', matrix[0, :])
print('Столбец 2:', matrix[:, 2])
```

Операции с элементами матриц

С элементами матриц можно делать всё то же, что и с элементами векторов. Две матрицы можно сложить, вычесть, умножить или поделить. Главное, чтобы действия производились поэлементно, а матрицы были одинакового размера. Результат операций — матрица того же размера.

```
import numpy as np

matrix1 = np.array([
    [1, 2],
    [3, 4]])

matrix2 = np.array([
    [5, 6],
    [7, 8]])

print(matrix1 + matrix2)
```

Матрицу можно умножить на число, прибавить его или вычесть: операция применяется к каждому элементу.

```
import numpy as np

matrix = np.array([
    [1, 2],
    [3, 4]])

print(matrix * 2)
print(matrix - 2)
```

Умножение матрицы на вектор

Чтобы понять, как матрица умножается на вектор, возьмём список строк. Каждая строка этого списка (матрицы) скалярно умножается на вектор, а полученные числа образуют новый вектор.

Например, матрицу A размера $m \times n$ умножим на вектор *b* (*n*-мерный). Произведением будет новый вектор *c*=Ab. Это m-мерный вектор, у которого i-я координата равна скалярному произведению *i*-й строки матрицы на *b*.

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{pmatrix}$$

$$\begin{aligned} c_1 &= a_{11} \times b_1 + a_{12} \times b_2 + \ldots + a_{1n} \times b_n \\ c_2 &= a_{21} \times b_1 + a_{22} \times b_2 + \ldots + a_{2n} \times b_n \\ &\vdots \\ c_m &= a_{m1} \times b_1 + a_{m2} \times b_2 + \ldots + a_{mn} \times b_n \end{aligned}$$

Выполним эту операцию в *NumPy*, вызовем знакомую вам функцию *np.dot()*.

```
import numpy as np

A = np.array([
    [1, 2, 3],
    [4, 5, 6]])

b = np.array([7, 8, 9])

print(np.dot(A, b))
print(A.dot(b))
```

Чтобы умножение было корректным, размер вектора должен быть равен ширине матрицы.

Транспонирование матриц

Транспонирование матрицы — это её «переворот» относительно главной диагонали матрицы, которая идёт из левого верхнего в правый нижний угол. При таком перевороте матрицы *A* размера $m \times n$ трансформируется в матрицу размера $n \times m$. То есть строки матрицы становятся её столбцами, а столбцы — строками. Транспонированная матрица обозначается верхним индексом **T**:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}^T = \begin{pmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{pmatrix}$$

В *NumPy* эта операция задаётся атрибутом *T*. Если нужно построить матрицу, начиная со списка столбцов, создайте матрицу и примените транспонирование:

```
import numpy as np

matrix = np.array([
    [1, 2],
    [4, -4],
    [0, 17]])

print("Транспонированная матрица")
print(matrix.T)
```

Умножим исходную матрицу на вектор длины 3:

```
vector = [2, 1, -1]
print(np.dot(matrix, vector))
```

Получили ошибку: размеры матрицы (3, 2) и вектора (3,) не согласованы. Вторая размерность матрицы не равна длине вектора. Чтобы умножение было корректным, транспонируем матрицу:

```
print(np.dot(matrix.T, vector))
```

Матричное умножение

При **матричном умножении** по двум матрицам строится третья. Она состоит из скалярных произведений строк первой матрицы на столбцы второй. Произведение *i*-й строки матрицы *A* (*Ai*) на *j*-й столбец матрицы *B* (*Bj*) равно матрице *Cij:*

$$c_{ij} = (A_i, B_j)$$

Умножение матрицы на матрицу возможно, если ширина первой матрицы *A* ($m \times n$) равна высоте второй матрицы *B* ($n \times r$). Тогда размер произведения этих матриц будет $m \times r$. Размерность *n* «схлопывается».

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1r} \\ b_{21} & b_{22} & \cdots & b_{2r} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nr} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1r} \\ c_{21} & c_{22} & \cdots & c_{2r} \\ \vdots & \vdots & \vdots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mr} \end{pmatrix}$$

$$\begin{aligned} c_{11} &= a_{11} \times b_{11} + a_{12} \times b_{21} + \ldots + a_{1n} \times b_{n1} \\ c_{12} &= a_{11} \times b_{12} + a_{12} \times b_{22} + \ldots + a_{1n} \times b_{n2} \\ &\vdots \\ c_{1r} &= a_{11} \times b_{1r} + a_{12} \times b_{2r} + \ldots + a_{1n} \times b_{nr} \end{aligned}$$

В *NumPy* матрицы *A* и *B* умножаются вызовом функций *np.dot(A, B)*, или *A.dot(B)*. Также этот вызов можно заменить знаком матричного умножения @:

```
import numpy as np

print(A.dot(B))
print(np.dot(A,B))
print(A @ B)
```

Результат умножения матриц зависит от порядка множителей. Если ширина первой матрицы не равна длине второй, то умножение невозможно.

```
matrix = np.array([
    [1, 2, 3],
    [-1, -2, -3]])

print(matrix @ matrix)
```

Произошла ошибка: размерности матриц не согласованы.

Умножение матрицы на себя возможно, только если она квадратная:

```
square_matrix = np.array([
    [1, 2, 3],
    [-1, -2, -3],
    [0, 0, 0]])

print(square_matrix @ square_matrix)
```