Матрица описывается количеством столбцов (columns), количеством строк (rows) и массивом данных (data). Хотя матрица является двумерным массивом, данные записаны в одномерном массиве:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = (1,2,3,4,5,6,7,8,9)$$

$$data[1 \cdot columns + 2] = 6$$

$$data[2 \cdot columns + 0] = 7$$

$$columns = rows = 3;$$

В классе определены следующие конструкторы:

- конструктор по умолчанию;
- конструктор с параметрами (размерность матрицы);
- конструктор с параметрами (размерность + массив данных);
- конструктор копирования;

Метод **file_read()** предназначен для чтения данных из файла, числа должны разделяться пробелом/знаком переноса. Данные записываются в матрицу размера (1, counter), где counter – кол-во чисел в файле, таким образом чтение происходит до конца файла.

Метод **file_write()** записывает содержимое матрицы в указанный файл с переносом строки после записи каждого элемента.

Метод **minor()** возвращает матрицу, полученную путем удаления столбца и строки, в которых находится указанный элемент:

$$minor(1,2) = \begin{pmatrix} 1 & 2 & \cancel{3} \\ \cancel{4} & \cancel{5} & \cancel{6} \\ 7 & 8 & \cancel{9} \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 7 & 8 \end{pmatrix}$$

Метод **deter()** возвращает детерминант матрицы, который высчитывается путем приведения матрицы к верхне-треугольному виду и последующим перемножением элементов на главной диагонали (метод Гаусса).

$$\begin{pmatrix} 1 & 3 & 6 \\ 2 & 4 & 4 \\ 3 & 3 & 3 \end{pmatrix} \to \begin{pmatrix} 1 & 3 & 6 \\ 0 & -2 & -8 \\ 0 & -6 & -15 \end{pmatrix} \to \begin{pmatrix} 1 & 3 & 6 \\ 0 & -2 & -8 \\ 0 & 0 & 9 \end{pmatrix}$$

$$\det = 1 \cdot (-2) \cdot 9 = -18$$

Оператор «!» используется для получения обратной матрицы методом Гаусса-Жордана. К исходной матрице справа приписывается единичная матрица той же размерности, затем, путем элементарных преобразований, левая часть (изначальная матрица) полученной совмещенной матрицы приводится к единичному виду. После всех преобразований правая часть, которая раньше представляла собой присоединенную единичную матрицу, будет представлять собой обратную матрицу.

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

$$\left(\begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ 3 & 4 & 0 & 1 \end{array}\right) \to \left(\begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ 0 & -2 & -3 & 1 \end{array}\right) \to \left(\begin{array}{cc|cc} 1 & 0 & -2 & 1 \\ 0 & -2 & -3 & 1 \end{array}\right) \to \left(\begin{array}{cc|cc} 1 & 0 & -2 & 1 \\ 0 & 1 & 3/2 & -1/2 \end{array}\right)$$

$$A^{-1} = \begin{pmatrix} -2 & 1 \\ 3/2 & -1/2 \end{pmatrix}$$

Метод **eye()** возвращает единичную матрицу той же размерности что и исходная.

Метод **merg_by_columns()** возвращает матрицу, состоящую из исходных матриц, склеенных вдоль строк:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}; B = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$$

$$A.merg\_by\_columns(B) = \begin{pmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \end{pmatrix}$$

Метод **merg_by_rows()** возвращает матрицу, состоящую из исходных матриц, склеенных вдоль столбцов:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}; B = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$$

$$A.merg\_by\_rows(B) = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{pmatrix}$$

Метод **shift()** производит сдвиг значений по строкам матрицы, при этом значение сдвига для каждого столбца задается отдельно. Удобен для реализации задержек для входов нейросети. Результат работы метода при массиве сдвигов s=[0, 1, 2]:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

$$A.shift(s) = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 7 \end{pmatrix}$$

Метод **cut_rows()** возвращает матрицу, состоящую из указанных строк изначальной матрицы. Необходимые строки задаются индексами начальной и конечной строки.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

$$A.cut\_rows(0,0) = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$$

$$A.cut\_rows(1,2) = \begin{pmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Метод **cut_columns()** аналогичен методу **cut_rows()** но применяется к столбцам матрицы:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

$$A.cut\_columns(0,0) = \begin{pmatrix} 1 \\ 4 \\ 7 \end{pmatrix}$$

$$A.cut\_columns(1,2) = \begin{pmatrix} 2 & 3 \\ 5 & 6 \\ 8 & 9 \end{pmatrix}$$

Метод **insert_columns()** создает матрицу на основе исходной, путем вставки в нее значений из другой матрицы, начиная с некоторого столбца:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} B = \begin{pmatrix} 7 \\ 7 \\ 7 \end{pmatrix}$$

$$A.insert\_columns(1,B) = \begin{pmatrix} 1 & 7 & 3 \\ 4 & 7 & 6 \\ 7 & 7 & 9 \end{pmatrix}$$

Метод **expand()** преобразует матрицу в одномерный вектор. Так как данные уже хранятся в одномерном массиве изменяется только размерность матрицы.