Linear Algebra: Matrices I

Glenn Bruns
CSUMB

Much of the material in these slides comes from Géron's notes:
https://github.com/ageron/handson-ml/blob/master/math_linear_algebra.ipynb

Learning outcomes

After this lecture you should be able to:

1. Define ‘matrix’
2. Define some special matrices:
   - square, diagonal, identity
3. Define and perform matrix operations
   - addition, multiplication, matrix multiplication

Matrix

A matrix is a rectangular array of scalars arranged in rows and columns.

Example:

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

- use upper case letters for matrices
- A has 2 rows and 3 columns
- the size of A is 2×3 (rows x columns)

Example: feature data

Points in an n-dimensional feature space:

$$\begin{array}{cc} & \begin{array}{ccc} \text{feature 1} & & \text{feature 3} \end{array} \\ \begin{array}{c} \text{point 1} \\ \text{point 2} \end{array} & \begin{pmatrix} 1.2 & -5.0 & 3.5 \\ 1.4 & -6.0 & 4.7 \end{pmatrix} \end{array}$$

If there are m points and n features, what is the size of the matrix?

Matrices in Python and NumPy

```
# you can use a list of Python lists
A = [ [10, 20, 30], [40, 50, 60] ]
# we will use NumPy
A = np.array([
        [10, 20, 30],
        [40, 50, 60]
    ])
# getting the size of a NumPy matrix
A.shape
# the size attribute gives the number of elements
A.size
```

!!

Indexing into matrices

```
A = np.array([
        [10, 20, 30],
        [40, 50, 60]
    ])
```

```
# simple indexing; rows then columns
A[1,2]        ?
# : in second position for all columns
A[1, :]       ?
# : in first position for all rows
A[:, 2]       ?
```

A[1, :] looks like a row vector, and A[:, 2] looks like a column vector, but in NumPy both are just 1D NumPy arrays.

```
# use slicing to get row 2 as a row vector
A[1:2, :]
# to get column 3 as a column vector
A[:, 2:3]
```

Matrix addition

Simply elementwise addition:

$$\begin{pmatrix} 1 & 0 & 0 \\ 6 & 2 & 0 \\ 7 & 8 & 3 \end{pmatrix} + \begin{pmatrix} 4 & 7 & 8 \\ 0 & 5 & 9 \\ 0 & 0 & 6 \end{pmatrix} = \begin{pmatrix} 5 & 7 & 8 \\ 6 & 7 & 9 \\ 7 & 8 & 9 \end{pmatrix}$$

Matrix addition is:

commutative

$$A + B = B + A$$

associative

$$A + (B + C) = (A + B) + C$$

In Numpy, just add Numpy matrices using '+'.

```
A = np.array([
    [1,0,0], [6,2,0], [7,8,3]])
B = np.array([
    [4,7,8], [0,5,9], [0,0,6]])
A + B
```

Multiplication by a scalar

This is easy:

$$3\begin{pmatrix}1 & 0 & 3\\4 & 6 & 8\\2 & 7 & 5\end{pmatrix} = \begin{pmatrix}3 & 0 & 9\\12 & 18 & 24\\6 & 21 & 15\end{pmatrix}$$

Commutative:

$$kA = Ak$$

Associative:

$$k_1(k_2A) = (k_1k_2)A$$

Distributes over matrix addition:

$$k(A + B) = kA + kB$$

Comparing vectors and matrices

vectors

$$u + w = w + u$$

$$u + (w + v) = (u + w) + v$$

$$(bc)u = b(cu)$$

$$c(u + w) = cu + cw$$

$$(b + c)u = bu + cu$$

matrices

$$A + B = B + A$$

$$A + (B + C) = (A + B) + C$$

$$(bc)A = b(cA)$$

$$c(A + B) = cA + cB$$

$$(b + c)A = bA + cA$$

b, c – scalars
u, v, w – vectors
A, B, C - matrices

Square, diagonal, identity matrices

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

square

(num rows = num cols)

$$\begin{pmatrix} 6 & 4 & 2 \\ 0 & 8 & 7 \\ 0 & 0 & 3 \end{pmatrix}$$

upper triangular

(square, and values below main diagonal all 0)

$$\begin{pmatrix} 6 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

diagonal

(both upper and lower triangular)

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

identity (written I)

(square, with 1 on diagonal and 0 off diagonal)

Matrix multiplication

This is trickier:

$$\begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 3 \\ 4 & 6 & 8 \\ 2 & 7 & 5 \end{pmatrix} = \begin{pmatrix} 15 & 33 & 34 \\ 13 & 19 & 30 \end{pmatrix}$$

Two things to remember:

1. # of cols in first matrix must equal # of rows in second matrix

   m x **n** **n** x p → m x p

2. Each element in the result comes from a dot product

$$\begin{pmatrix} \boxed{1 \quad 2 \quad 3} \\ 3 \quad 2 \quad 1 \end{pmatrix} \begin{pmatrix} 1 & \boxed{0} & 3 \\ 4 & \boxed{6} & 8 \\ 2 & \boxed{7} & 5 \end{pmatrix} = \begin{pmatrix} 15 & \boxed{33} & 34 \\ 13 & 19 & 30 \end{pmatrix}$$

Memory aid

To get the result value in the 1st row, 2nd column:

use the 1st row of matrix A, 2nd column of matrix B

$$\begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 3 \\ 4 & 6 & 8 \\ 2 & 7 & 5 \end{pmatrix} = \begin{pmatrix} 15 & 33 & 34 \\ 13 & 19 & 30 \end{pmatrix}$$

Matrix multiplication exercises

$$\begin{pmatrix} 1 & 2 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 3 & 0 \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} =$$

Properties of matrix multiplication

Let A, B, C be matrices

1. Is it true that AB = BA? (commutativity)

   try it: $\begin{pmatrix} 3 & 4 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 6 & 2 \\ 3 & 2 \end{pmatrix}$

2. Is it true that A(BC) = (AB)C? (associativity)

   yes

3. Is it true that A(B + C) = AB + AC? (left distributivity)

   yes

Matrix multiplication exercises

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} =$$

Matrix multiplication in NumPy

code:

```
A = np.array([
        [10, 20, 30],
        [40, 50, 60]
    ])
D = np.array([
        [ 2,  3,  5,  7],
        [11, 13, 17, 19],
        [23, 29, 31, 37]
    ])
E = A.dot(D)
```

output in Spyder:

```
In [5]: E = A.dot(D)
   ...: E
Out[5]:
array([[ 930, 1160, 1320, 1560],
       [2010, 2510, 2910, 3450]])
```

Summary

1. Matrices are rectangular arrays of scalars
2. Some special matrices: square, diagonal, identity
3. Matrix operations: addition, multiplication by scalar, multiplication
4. Matrix and matrix operations in NumPy