06 Matrices

Class Survey

Fill out the survey at this link! :)

https://forms.gle/E7sQc2URLSvkJcqu7

Unit 1: Vectors, Book ILA Ch. 1-5

Unit 2: Matrices, Book ILA Ch. 6-11 + Book IMC Ch. 2

- ***06 Matrices***
- 07 Linear Equations
- 08 Linear Dynamical Systems
- 09 Matrix Multiplication
- 10 Matrix Inverse

Unit 3: Least Squares, Book ILA Ch. 12-14 + Book IMC Ch. 8

Unit 4: Eigen-decomposition, Book IMC Ch. 10, 12, 19

Outline: 06 Matrices

- **Matrices**
- Matrix-vector multiplication
- Examples

Matrices

Definition: A matrix is a rectangular array of numbers, e.g.:

$$A = \begin{bmatrix} 0 & 1 & -2.3 \\ 1.3 & 4 & -0.1 \end{bmatrix}$$

- Its size, or shape, is: (row dimension) x (column dimension).
  - Example: Matrix above has size 2 x 3.
- Its elements are called: entries, coefficients.
- $A_{i,j}$ refers to element at ith row and jth column in matrix A.
  - i is the row index and j is the column index.

In Python, we use `numpy` and `np.array` to build matrices. The shape of the matrix can be accessed via the function `shape`.

In [7]:

```
import numpy as np

A = np.array([
    [0, 1, -2.3],
    [1.3, 4, -0.1]
])
print(np.shape(A)); print(A.shape)
v = np.array([1, 2, 3, 4]); print(len(v))
```

```
(2, 3)
(2, 3)
4
```

In Python, we can access the elements of the matrix.

In [18]:

```
v = [1.1, 2.2, 3.3, 4.3]; #print(len(v))

for v_index in range(len(v)):
    print(v[v_index])

for v_value in v:
    print(v_value)
```

```
1.1
2.2
3.3
4.3
```

In [29]:

```
print(A); print(A[0][2]); print(A[0, 2])
# Implement code that prints all entries of the matrix A
print("Begin for loop:")
print(A.shape)
n_rows = A.shape[0]; n_columns = A.shape[1]
n_rows, n_columns = A.shape

for i in range(n_rows):
    for j in range(n_columns):
        print(A[i, j])
```

```
[[ 0.   1.  -2.3]
 [ 1.3  4.  -0.1]]
-2.3
-2.3
Begin for loop:
(2, 3)
0.0
1.0
-2.3
```

```
1.3
4.0
-0.1
```

Sizes/Shapes of Matrices

Definitions: A m x n matrix A is:

- tall if m > n,
- wide if m < n,
- square if m = n.

In [3]:

```python
import numpy as np

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

Out[3]:

```
(2, 3)
```

Matrices, Vectors and Scalars

Definitions:

- A 1 x 1 matrix is a number or scalar.
- A n x 1 matrix is an n-vector.
- A 1 x n matrix is a n-row-vector.

Starting now, we will distinguish vectors and row vectors.

Columns and rows of a matrix

Notations: Take A a $m \times n$ matrix with entries A_{ij} for $i = 1, \ldots, m$ and $j = 1..., n$.

- Its jth column is the m-vector:

$$\begin{bmatrix} A_{1j} \\ \vdots \\ A_{mj} \end{bmatrix}$$

- Its ith row is the n-row-vector: $[A_{i1}, \ldots, A_{in}]$.

Slices of a matrix

Definition The slice of matrix $A_{p:q,r:s}$ is the matrix:

$$\begin{bmatrix} A_{pr} & A_{p,r+1} & \dots & A_{ps} \\ \dots & \dots & \dots & \dots \\ A_{qr} & A_{q,r+1} & \dots & A_{qs} \end{bmatrix}$$

In Python, we can extract rows, columns and slices:

In [20]:

```
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]); print("Matrix:"); print(A)
# print(A[0, :]); print(A[:, 1]); print(A[0, 1])

# 1, 2, 4, 5
A[0:(1+1), 1:(2+1)]
```

```
Matrix:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Out[20]:

```
array([[2, 3],
       [5, 6]])
```

Block matrices

Definition: A matrix A composed from other matrices is called a block matrix:

$$A = \begin{bmatrix} B & C \\ D & E \end{bmatrix}$$

where B, C, D, E are called submatrices or blocks of A.

Column/row representations

Notations: Take A a $m \times n$ matrix with entries A_{ij} for $i = 1, \dots, m$ and $j = 1..., n$.

- A is the block matrix of its columns $a_1, \dots, a_n$:
  - $A = [a_1 \dots a_n]$
- A is the block matrix of its rows $b_1, \dots, b_m$:
  - $A = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$.

Examples in ECE and beyond

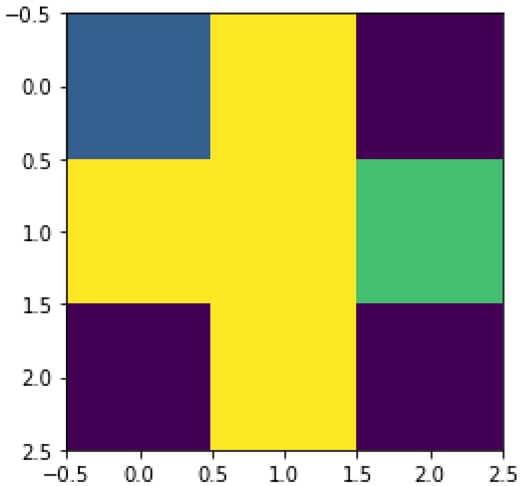
- Images: A_{ij} is intensity value at i, j.

In Python, we can plot an image represented by a matrix.

In [32]:

```
A = np.array([[0.3, 1, 0],[1, 1, 0.7],[0, 1, 0]]); A
import matplotlib.pyplot as plt
```

```
plt.imshow(A, cmap="viridis");
#plt.axis("off")
```



Examples in ECE and beyond

- Weather: A_{ij} is rainfall data at location i on day j.
- Finances: A_{ij} is the return of asset i in period j.

Exercise: In each of these, what do the rows and columns mean?

Special Matrices

Definition: The $m \times n$ zero matrix (resp. ones-matrix) is the matrix with all entries equal to 0 (resp. to 1).

Definition: The identity matrix I is the square matrix with $I_{ii} = 1$ and $I_{ij} = 0$ if $i \neq j$, for example:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

In Python:

In [44]:

```
zero_vec = np.zeros(4); print(zero_vec)
ones_vec = np.ones(5); print(ones_vec)
zero_mat = np.zeros((2, 3)); print(zero_mat) # 2 x 3
ones_mat = np.ones((2, 4)); print(ones_mat)  #2x4

identity_mat = np.identity(4); identity_mat
```

```
[0. 0. 0. 0.]
[1. 1. 1. 1. 1.]
[[0. 0. 0.]
 [0. 0. 0.]]
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

Out[44]:
```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

Diagonal Matrices

Definition: A diagonal matrix A is a square matrix with $A_{ij} = 0$ for $i \neq j$.

- diag$(a_1, \ldots, a_n)$ denotes the diagonal matrix with $A_{ii} = a_i$:

$$\text{diag}(0.2, -3, 1.2) = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & 1.2 \end{bmatrix}$$

In Python:

In [47]:
```
np.diag(np.ones(7))
```

Out[47]:
```
array([[1., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 0., 1.]])
```

Triangular Matrices

Definition: A lower triangular matrix A is a matrix such that $A_{ij} = 0$ for $i < j$. An upper triangular matrix A is a matrix such that $A_{ij} = 0$ for $i > j$.

Example: $\begin{bmatrix} 0.2 & 1.2 & 10 \\ 0 & -3 & 0 \\ 0 & 0 & 1.2 \end{bmatrix}$ (upper-triangular)

Transpose

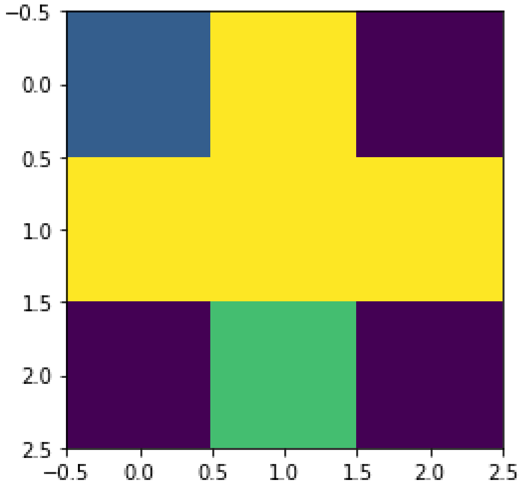
Definition: The transpose of an $m \times n$ matrix A is written A^T and is defined by:

$$(A^T)_{ij} = A_{ji}, \quad i = 1, \ldots, n \quad j = 1, \ldots, m$$

Example: $\begin{bmatrix} 0.2 & 1.2 & 10 \\ 0 & -3 & 0 \end{bmatrix}^T = \begin{bmatrix} 0.2 & 0 \\ 1.2 & -3 \\ 10 & 0 \end{bmatrix}$

In Python:

In [57]:
```
#print(A); print(A.T)
plt.imshow(A.T);
```



Addition, Substraction and Scalar Multiplication

Just like vectors:

- we can add or subtract matrices of the same size,
- we can multiply a matrix by a scalar.

Property: The transpose verifies:

- $(A^T)^T = A$.
- $(A+B)^T = A^T + B^T$.

Matrix norm

Definition: For a $m \times n$ matrix A, we define the matrix norm as:

$$||A|| = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}^2}.$$

Remark: This definition agrees with the definition of norm of vectors when $n = 1$ or $m = 1$.

Exercise: Compute the matrix norm of $A = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \end{bmatrix}$.

In Python, there are many ways to find the norm of a matrix.

In [23]:

```
import numpy as np

A = np.array([[-1, 0., 2.], [0., 1., 0.]])
print(A); print(np.linalg.norm(A));
# print(A ** 2); print(np.sqrt(np.sum(A**2)))

norm = 0
n_rows, n_cols = A.shape
for i in range(n_rows):
```

```
        for j in range(n_cols):
            norm = norm + A[i, j] ** 2
norm = np.sqrt(norm); print(norm)
```

```
[[-1.  0.  2.]
 [ 0.  1.  0.]]
2.449489742783178
2.449489742783178
```

Remark: As opposed to vectors which had 1 axis, matrices have several axes. Observe below how `numpy` functions adapt to the case of several axes.

In [27]:
```
print(A)
print(np.sum(A))
print(np.sum(A, axis=0))
print(np.sum(A, axis=1))
```

```
[[-1.  0.  2.]
 [ 0.  1.  0.]]
2.0
[-1.  1.  2.]
[1. 1.]
```

Distance between two matrices

Definition: The distance between two matrices A and B is defined as:

$$dist(A, B) = ||A - B||.$$

Remark: This means that the clustering algorithm, which only needs a notion of "distance", works on the matrices.

Outline: 06 Matrices

- Matrices
- **Matrix-vector multiplication**
- Examples

Matrix-vector multiplication

Definition: The matrix-vector multiplication y of $m \times n$ matrix A and n-vector x is denoted $y = Ax$ and is defined as:

$$y_i = A_{i1}x_1 + \ldots + A_{in}x_n, \quad i = 1, \ldots, m$$

Exercise: Let I be the $n \times n$ identity matrix and x an n-vector. Compute Ix.

Exercise: Let 0_n be the $n \times n$ zeroes matrix and x an n-vector. Compute $0_n x$.

Exercise: Compute the matrix-vector multiplication of:

$$A = \begin{bmatrix} 0 & 2 & -1 \\ -2 & 1 & 1 \end{bmatrix}; \quad v = \begin{bmatrix} 2 \\ 1 \\ -1 \end{bmatrix}$$

In Python:

In [31]:

```
print(A)
x = np.array([1, 2, 3]); print(x)

print(np.matmul(A, x))
print(A @ x)
```

```
[[-1.  0.  2.]
 [ 0.  1.  0.]]
[1 2 3]
[5. 2.]
[5. 2.]
```

Exercise: Take $m \times n$ matrix A, and one-hot vector e_i for some i in $1, \ldots, m$. Compute Ae_i.

Exercise: Take $m \times n$ matrix A, and ones vector 1_n. Compute $A1_n$.

- Matrices
- Matrix-vector multiplication
- **Examples**

Math operations as matrices

Example: The $(n-1) \times n$ difference matrix:

$$D = \begin{bmatrix} -1 & 1 & 0 & \ldots & 0 & 0 \\ 0 & -1 & 1 & \ldots & 0 & 0 \\ \\ 0 & 0 & 0 & \ldots & -1 & 1 \end{bmatrix}$$

allows us to compute the $(n-1)$ vector of differences between consecutive entries of x:

$$Dx = \begin{bmatrix} x_2 - x_1 \\ x_3 - x_2 \\ \vdots \\ x_n - x_{n-1} \end{bmatrix}$$

In Python, let be given a vector listing the days at which earthquakes have happened in California. Compute the vector giving the number of days in-between earthquakes.

In [43]:

```
earthquake_days = np.array([12, 45, 78])
D = np.array([
    [-1, 1, 0],
    [0, -1, 1]
])
```

```
print(D @ earthquake_days)
print(45 - 12)
print(78 - 45)
```

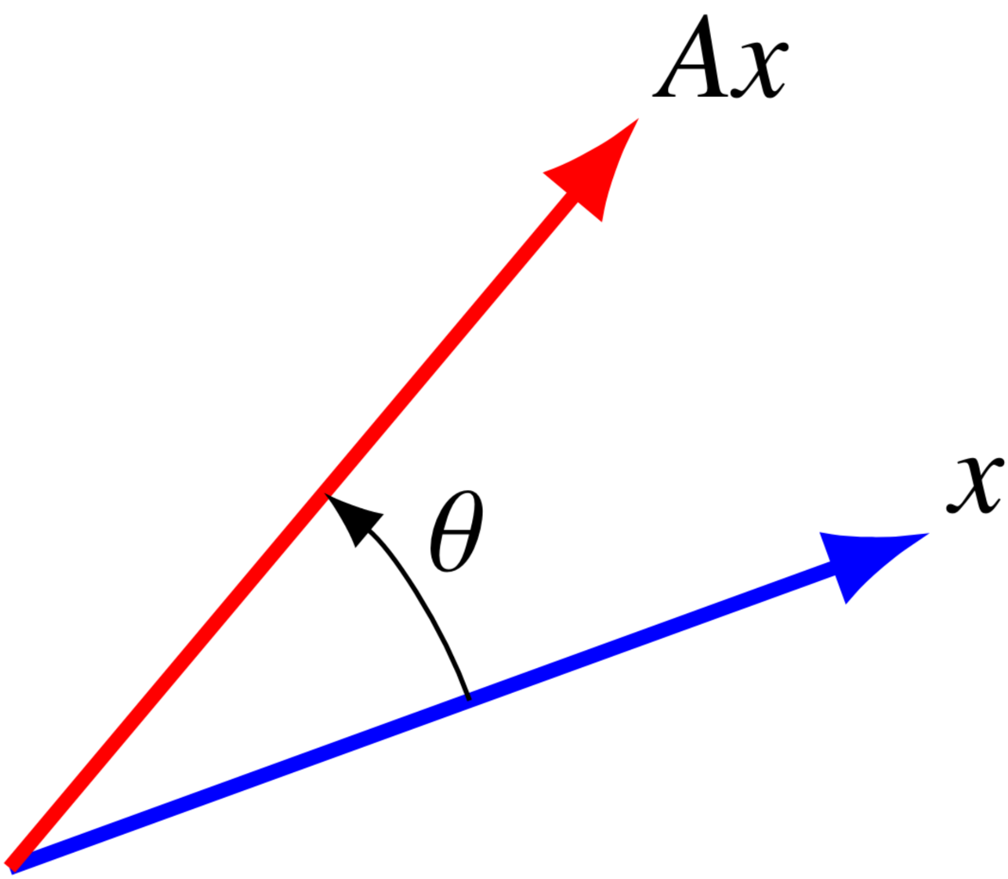
```
[33 33]
33
33
```

Geometric transformations as matrices

Many geometric transformations and mappings of 2-D and 3-D vectors can be represented as matrices A. They can transform vectors x through matrix-vector multiplication $y = Ax$.

Example: For example, the rotation by angle θ can rotate the $2D$-vector x through the rotation matrix R_θ:

$$R_\theta = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}, \quad y = R_\theta x$$

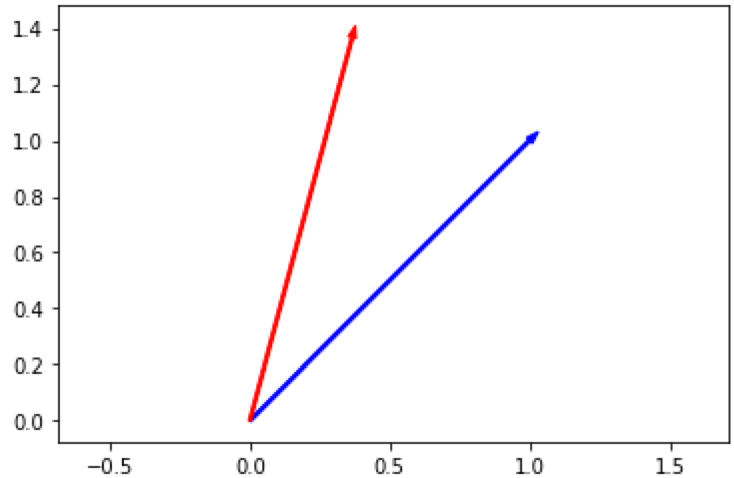


In Python:

In [53]:
```
x = np.array([1, 1])
import matplotlib.pyplot as plt

plt.arrow(0, 0, x[0], x[1], width=0.01, color="blue");
```

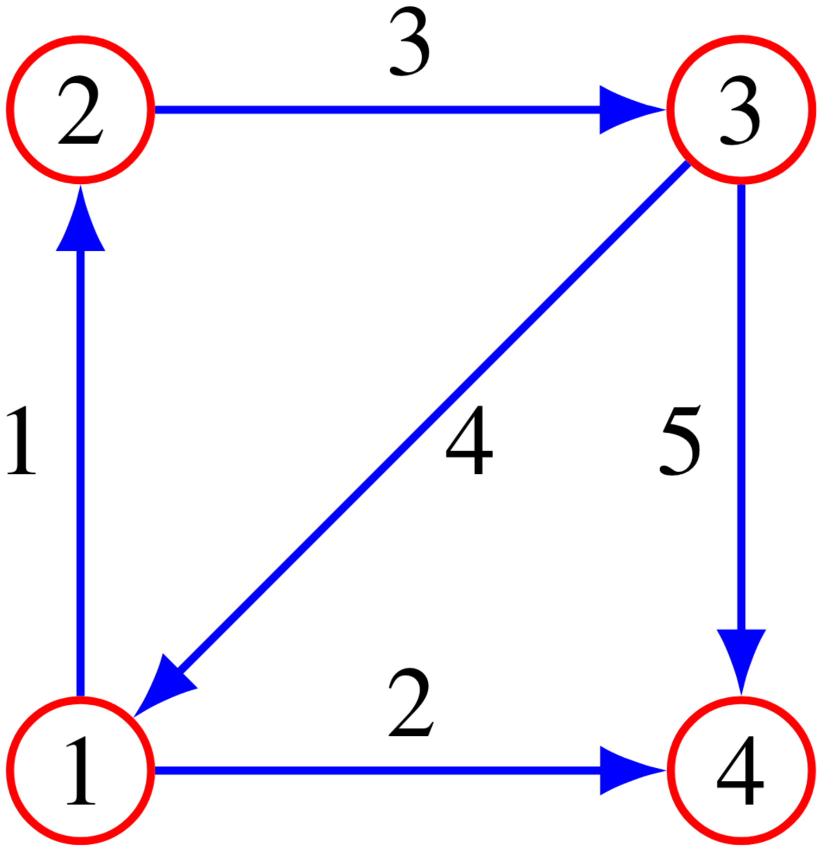
```
theta = np.pi / 6
R = np.array([[np.cos(theta), - np.sin(theta)], [np.sin(theta), np.cos(theta)]])
y = R @ x
plt.arrow(0, 0, y[0], y[1], width=0.01, color="red")
plt.axis("equal");
```



(Social) graphs as matrices

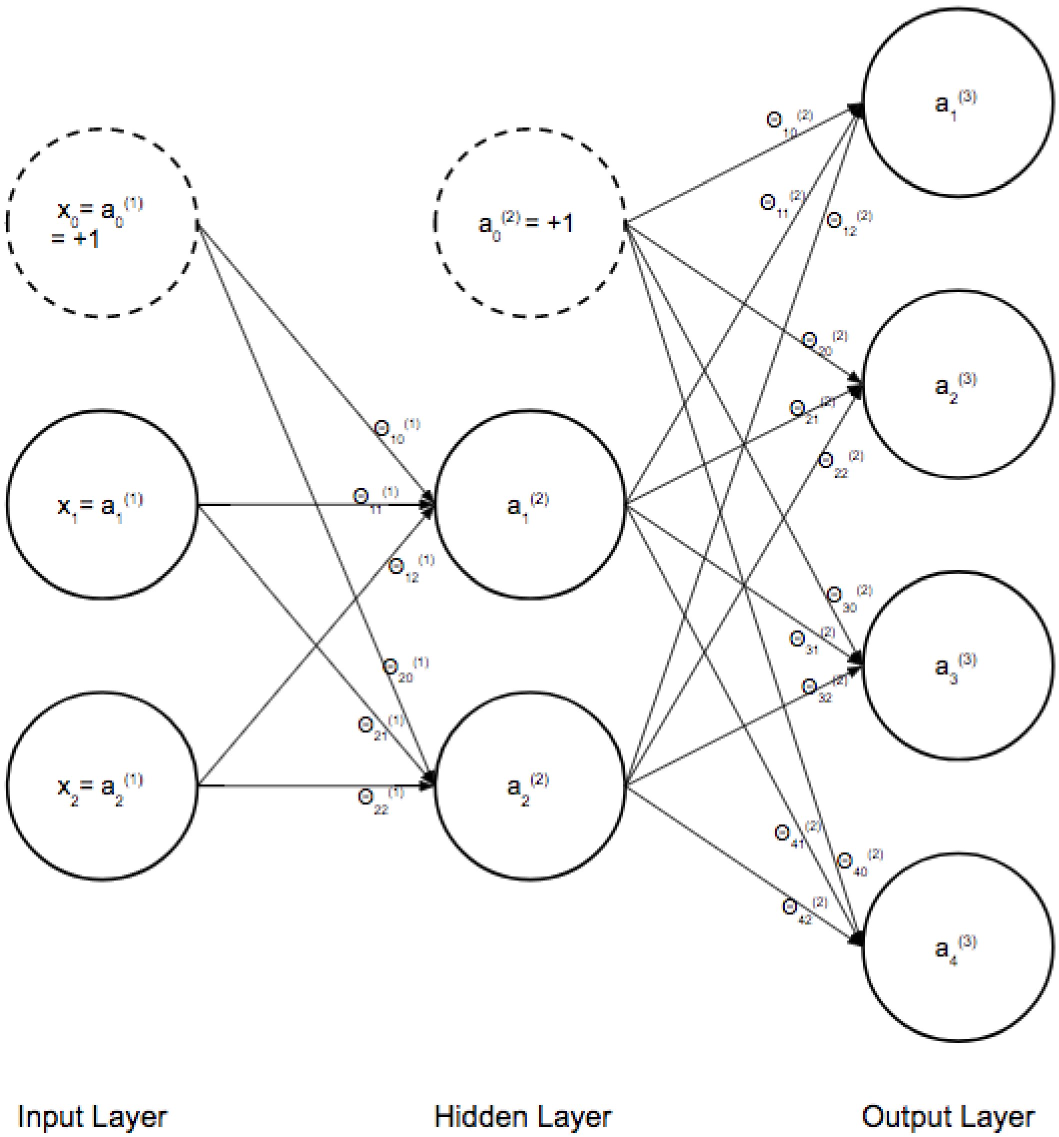
Any (social network) graph with n nodes can be represented by $n \times n$ matrix A, called adjacency matrix and defined as:

$$A_{ij} = \begin{cases} 1 & \text{if node } i \text{ points to node } j \\ 0 & \text{otherwise} \end{cases}$$



Exercise: Compute the adjacency matrix associated to this graph.

Deep Learning



By writing $a^{(1)}$ the vector of the first layer, and $a^{(2)}$ the vector of the second layer, we see that:

$$a^{(2)} = Wa^{(1)},$$

where W_{ij} is the weight on the edge going from the jth element of the first layer $a_j^{(1)}$ to the ith element of the second layer $a_i^{(2)}$.

In Python, we can implement the layer of a neural network.

In [54]:
```
def simple_nn(input_x, weights):
    return weights @ input_x
```

Outline: 06 Matrices

- Matrices
- Matrix-vector multiplication
- Examples

Resources: ILA, Ch. 6, 7.