CSC 461: Machine Learning

Fall 2024

Preliminaries

Prof. Marco Alvarez, Computer Science
University of Rhode Island

Linear algebra

Scalars and vectors

- Scalars
  - integers, real numbers, rational numbers, etc.
  - usually denoted by lowercase letters
- Vectors
  - 1-D array of elements (scalars)

$$\mathbf{x} \in \mathbb{R}^n$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Norms

- Functions that measure the **magnitude** ("length") of a vector
  - **strictly positive**, except for the zero vector
  - think about the distance between zero and the point represented by the vector

$$f(\mathbf{x}) = 0 \Rightarrow \mathbf{x} = \mathbf{0}$$
$$f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y}) \text{ (triangle inequality)}$$
$$\forall \alpha \in \mathbb{R}, f(\alpha \mathbf{x}) = |\alpha| f(\mathbf{x})$$

Norms

$$\ell_1\text{-norm:} \qquad \|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$$

$$\ell_2\text{-norm:} \qquad \|\mathbf{x}\|_2 = \left(\sum_{i=1}^{n} x_i^2\right)^{\frac{1}{2}}$$

$$\text{max norm:} \qquad \|\mathbf{x}\|_\infty = \max_i |\mathbf{x}_i|$$

Matrices and tensors

- Matrices
  - 2-D array of elements

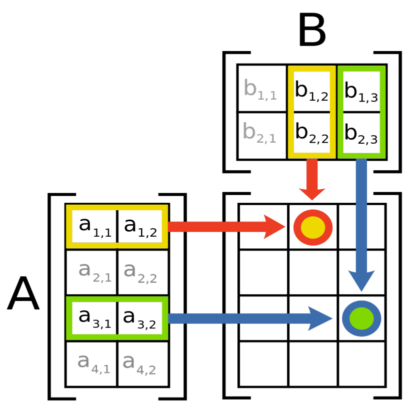
$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}_{m\times n} \qquad A \in \mathbb{R}^{m\times n}$$

- Tensors
  - homogeneous arrays that may have **zero or more** dimensions

Matrix multiplication

$$C = AB$$

$$C_{i,j} = \sum_k A_{i,k} B_{k,j}$$




```
Number of columns in A must be equal to the
number of rows in B
```

https://en.wikipedia.org/wiki/Matrix_multiplication

Dot product

$$\begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$\mathbf{x}^T\mathbf{y} = \sum_{i=1}^{n} x_i y_i$$

$$\mathbf{x}^T\mathbf{y} = \|\mathbf{x}\|\|\mathbf{y}\| \cos\theta$$

$$\mathbf{x}^T\mathbf{y} \in \mathbb{R}$$

Matrix transpose

$$\boldsymbol{A} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} \Rightarrow \boldsymbol{A}^\top = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix}$$

$$\left(A^T\right)_{i,j} = A_{j,i} \qquad (AB)^T = B^T A^T$$

Identity matrix

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

$$\forall x \in \mathbb{R}^n, I_n x = x$$

$$A^{-1}A = I_n$$

Special matrices and vectors

- Unit vector

$$\|\mathbf{x}\|_2 = 1$$

- Symmetric matrix

$$A = A^T$$

- Orthogonal matrix

$$A^T A = AA^T = I$$
$$A^{-1} = A^T$$

Python and Tensors

Python basics (must know)

- Basic data types
  - booleans, integers, floating point values, strings
- Control flow
- Built-in data structures
  - lists, tuples, dictionaries, sets
  - iterators
- Functions
  - functions can be assigned, passed, returned, and stored
  - lambda functions (inline and anonymous)
- Classes

Numpy

- Library for scientific computing
  - provides a **high-performance** multidimensional array object and routines for fast operations on these arrays
- The `ndarray` object encapsulates n-dimensional arrays of homogeneous data types
  - many operations performed as “compiled code” for higher performance
  - have a fixed size at creation
    - changing the size of an `ndarray` will create a new array and delete the original

```
n = 100000000
array1 = np.random.rand(n)
array2 = np.random.rand(n)
```

```
def dot_python(x, y):
    sum = 0
    for i in range(len(x)):
        sum += x[i] * y[i]
    return sum

def dot_numpy(x, y):
    return np.dot(x, y)
```

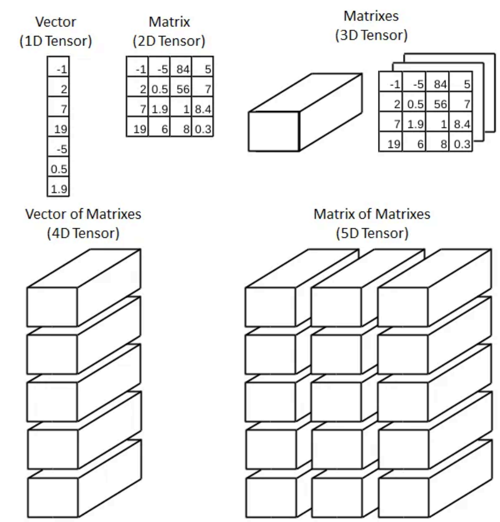
```
time_taken = timeit.timeit(lambda: dot_numpy(array1, array2), number=1)
print(f'Numpy Time: {time_taken} seconds')
```

```
Numpy Time: 0.05994947799996453 seconds
```

```
array1 = array1.tolist()
array2 = array2.tolist()
time_taken = timeit.timeit(lambda: dot_python(array1, array2), number=1)
print(f'Python Time: {time_taken} seconds')
```

```
Python Time: 8.325287125999978 seconds
```


Tensors



https://towardsdatascience.com/deep-learning-introduction-to-tensors-tensorflow-36ce3663528f

The following figures are from “NumPy Illustrated: The Visual Guide to NumPy” by Lev Maximov

https://betterprogramming.pub/numpy-illustrated-the-visual-guide-to-numpy-3b1d4976de1d

Elegant code

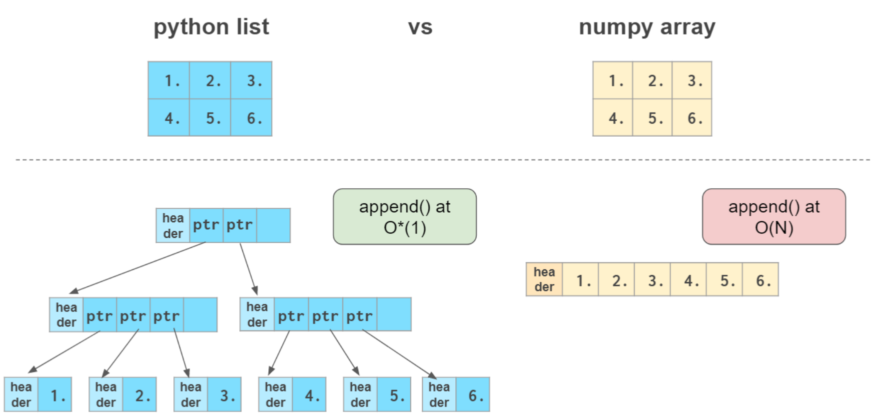
```
In [3]: a = [1, 2, 3]
        [q*2 for q in a]
Out[3]: [2, 4, 6]
```

```
In [4]: a = np.array([1, 2, 3])
        a * 2
Out[4]: array([2, 4, 6])
```

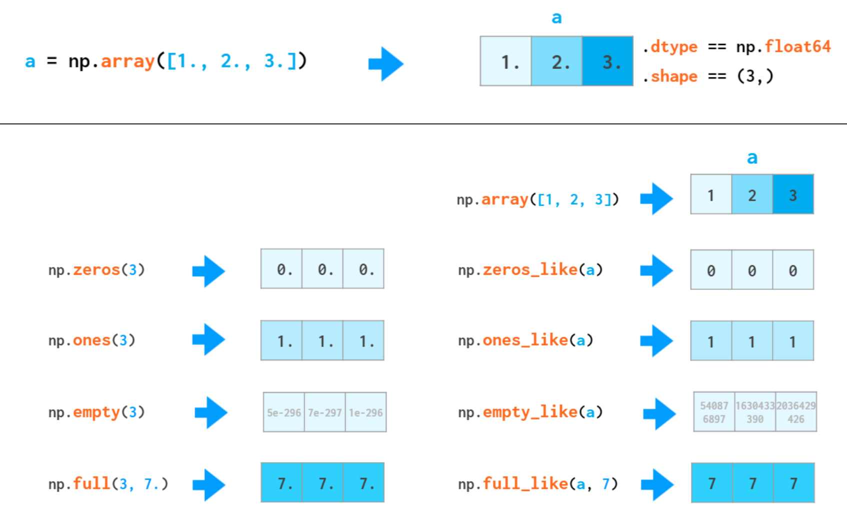
```
In [1]: a = [1, 2, 3]
        b = [4, 5, 6]
        [q+r for q, r in zip(a, b)]
Out[1]: [5, 7, 9]
```

```
In [2]: a = np.array([1, 2, 3])
        b = np.array([4, 5, 6])
        a + b
Out[2]: array([5, 7, 9])
```

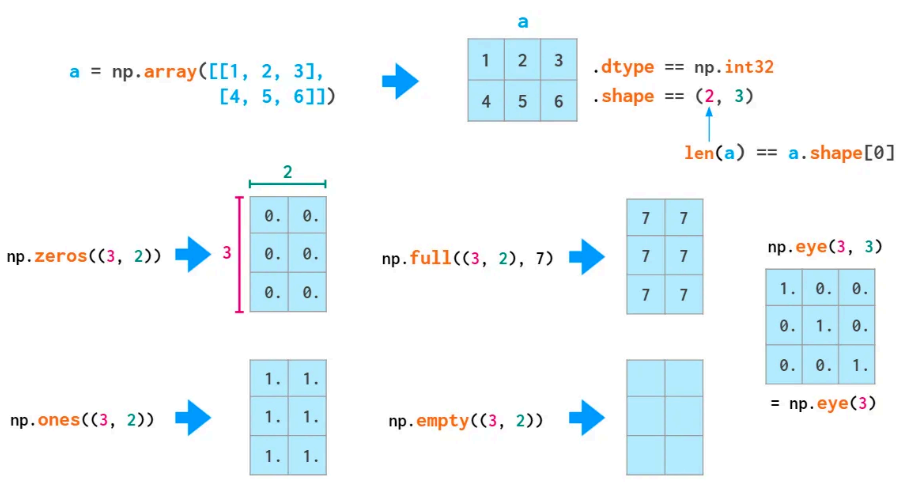
Homogeneous and fixed-length



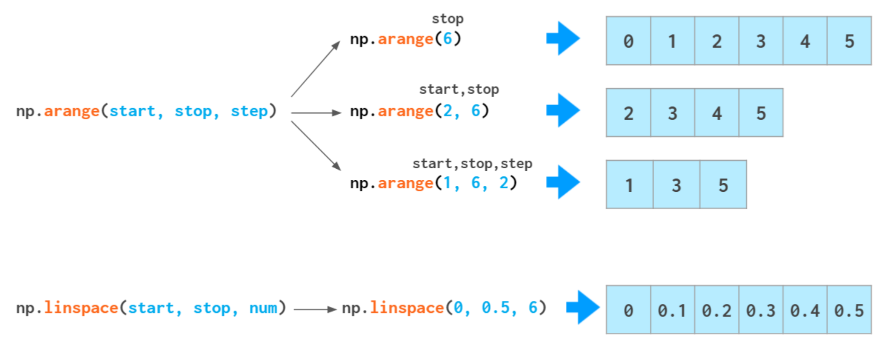
Creating numpy arrays



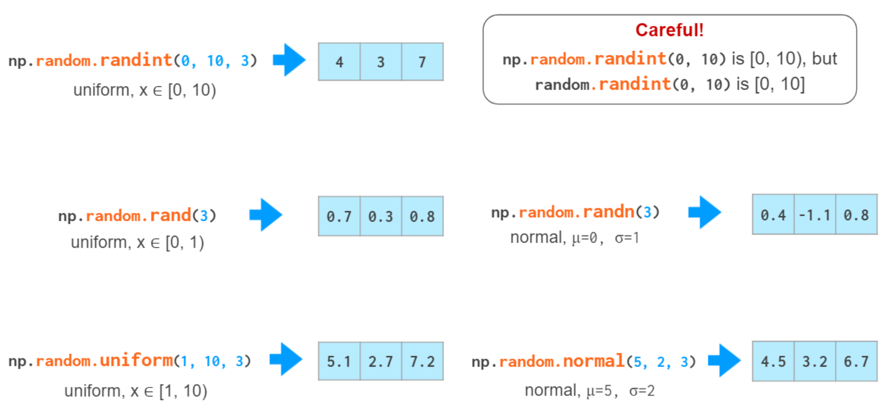
Creating numpy arrays



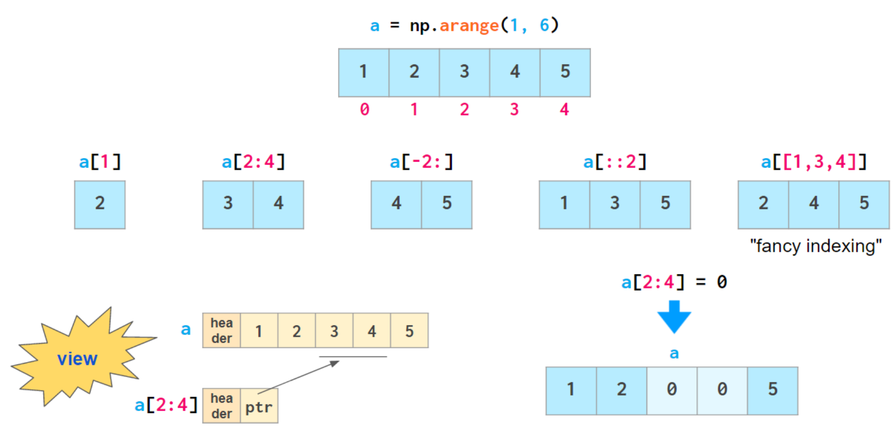
Creating numpy arrays



Creating numpy arrays



Indexing



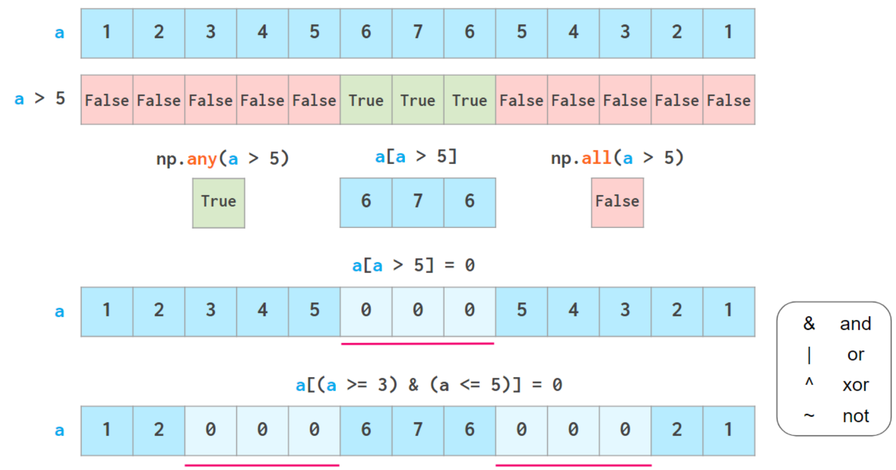
Careful when making copies

python list	vs	numpy array

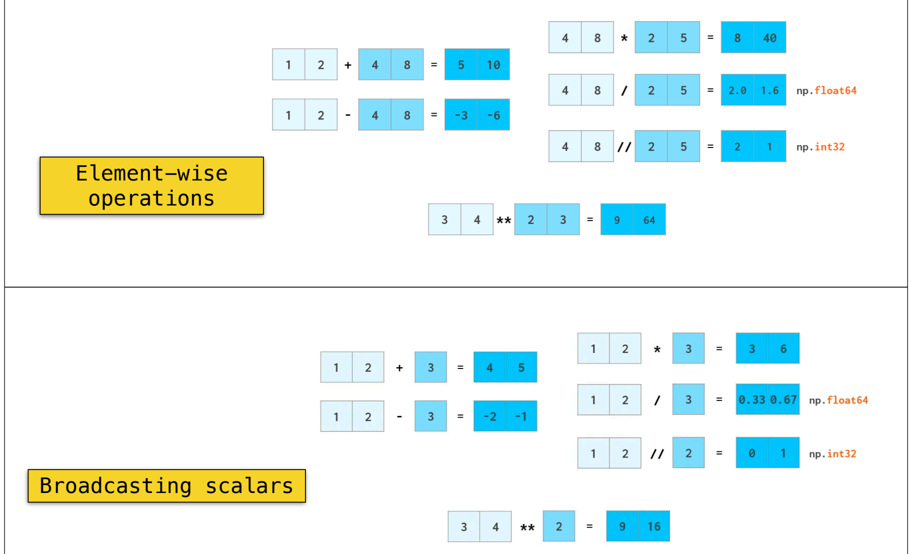
```
a = [1, 2, 3]
b = a          # no copy
c = a[:]       # copy
d = a.copy()   # copy
```

```
a = np.array([1, 2, 3])
b = a          # no copy
c = a[:]       # no copy!!!
d = a.copy()   # copy
```

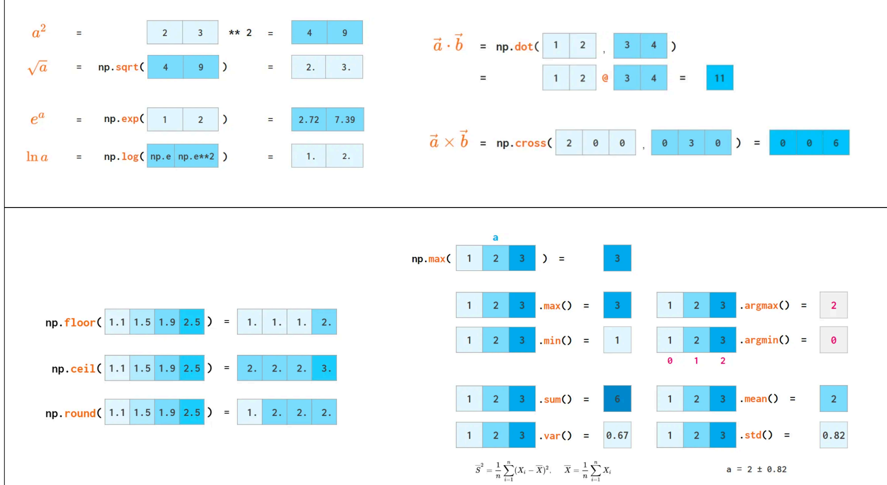
Boolean indexing



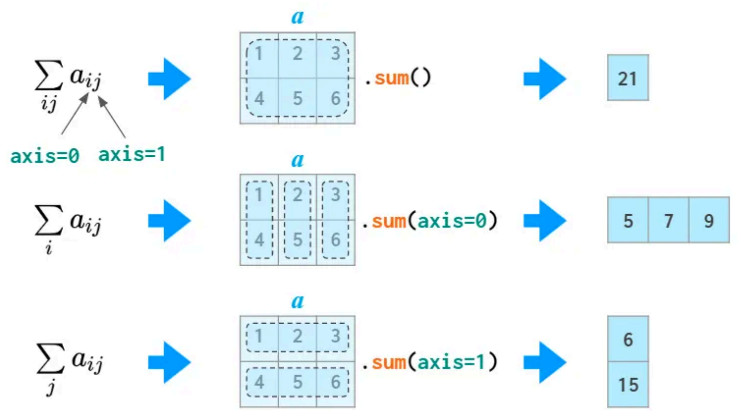
Vector operations



Math functions

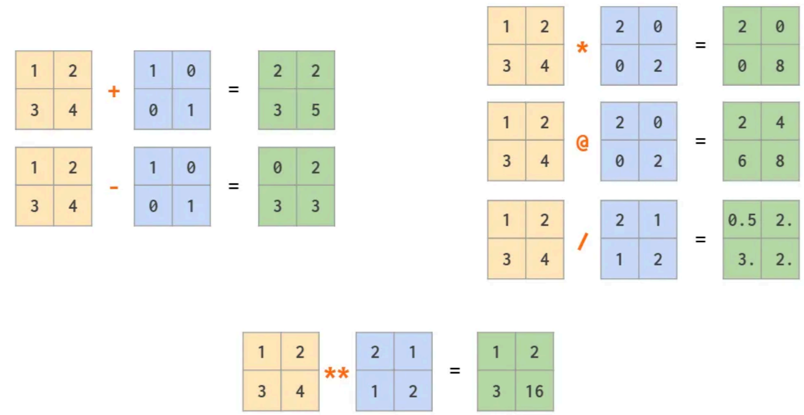


The axis



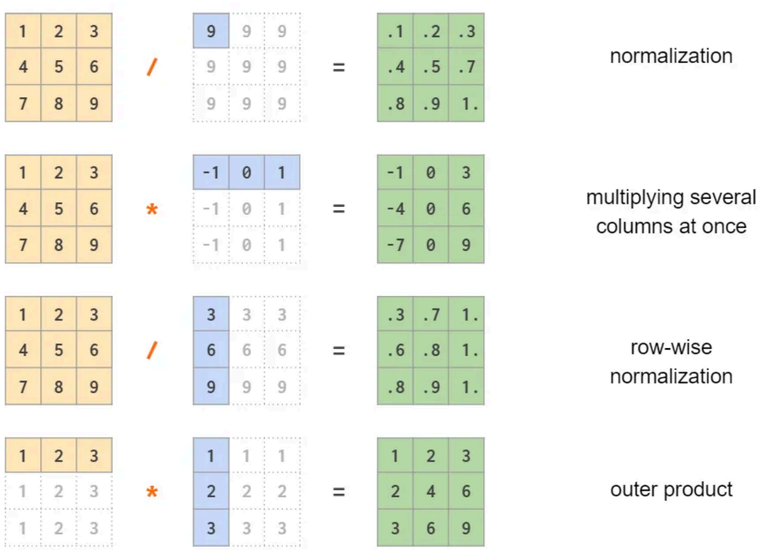
By specifying the **axis** parameter you can apply an operation along the specified axis of an array

Matrix operations



Not every operator or operation in NumPy is element-wise

Broadcasting



General broadcasting rules

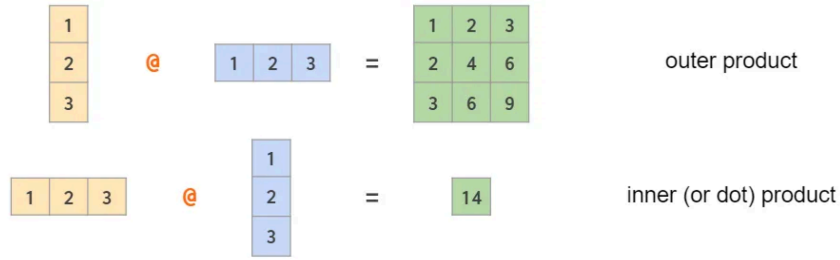
- Numpy follows a <u>set of rules</u> to determine how the smaller array is **broadcasted** to match the shape of the larger array
  - if the arrays have a different number of dimensions, **<u>pad the smaller array's shape</u>** with ones on its left
  - start with the rightmost dimension and **<u>work backwards</u>**
  - two dimensions are **<u>compatible</u>** if they are equal, or one of them is 1
    - otherwise throw a ValueError exception: "operands could not be broadcast together"
- Resulting array will have the same number of dimensions as the input array with the greatest number of dimensions
  - the size of each dimension is the largest size of the corresponding dimension among the input arrays

Practice

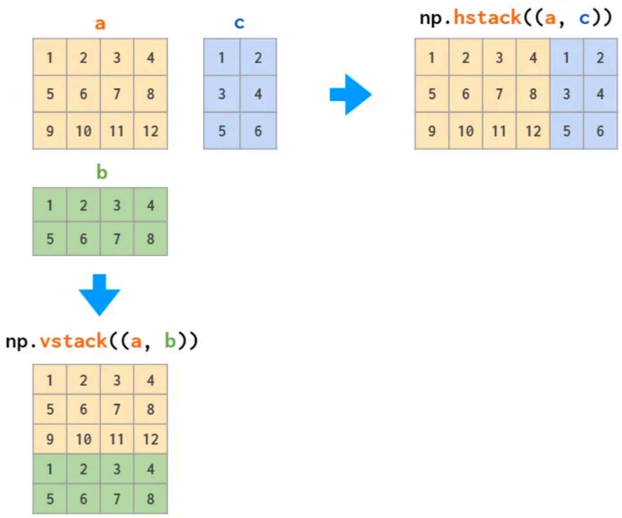
Arrays	Result
8 x 1 x 6 x 1 7 x 1 x 5	8 x 7 x 6 x 5
5 x 4 1	5 x 4
5 x 4 4	5 x 4
15 x 3 x 5 15 x 1 x 5	15 x 3 x 5
15 x 3 x 5 3 x 5	15 x 3 x 5
15 x 3 x 5 3 x 1	15 x 3 x 5
3 4	mismatch
2 x 1 8 x 4 x 3	mismatch

Arrays	Result
4 x 3 3	
5 x 4 x 3 1 x 3	
8 x 1 x 6 7 x 1	
5 x 2 5	
3 x 1 3 x 5	
2 x 4 x 1 8	
6 x 3 3 x 1	
3 x 5 x 2 3	

Must know your shapes …



Concatenation



Practice

- Task
  - given n data points, find the nearest neighbor to a query data point q
- Input:
  - input vector (vector_dim)
  - data (num_vectors, vector_dim)
- Output:
  - nearest_neighbor_idx
- Hint:
  - use euclidean distance for distance calculations
  - use vectorized and broadcasting operations

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$