

Linear Algebra Introduction

Objectives

By the end of this lesson, you will be able to:

- ❑ Define the terms scalar, vector, and matrix
- ❑ Compare and contrast operations that calculate length and distance of vectors
- ❑ Perform vector dot product multiplication and matrix multiplication
- ❑ Describe three techniques data scientists use to featurize real-world data
- ❑ Use NumPy to perform linear algebra on scalars, vectors, and matrices

Scalars

- A **scalar** is quantity that only has a magnitude (not a direction)
 - I did **2** loads of laundry.
 - The check was **\$16.72**.
 - The answer is **-0.356**.
- Typographically represented as a lowercase variable:
 - ρ
 - t
 - x

Vectors

A **vector** is a quantity that has a magnitude and a direction. You can think of axes (columns) of data as directions that define the vector space of the data, and the values inside as the magnitude in each of those directions.

Bold, lowercase: **x**

An **array** is an ordered sequence of values.

A vector **is** an array.

Vectors

A **vector** is a quantity that has a magnitude and a direction. You can think of axes (columns) of data as directions that define the vector space of the data, and the values inside as the magnitude in each of those directions.

Bold, lowercase: **x**

Example:

- We were driving **30 mph** going **North**.

Speed North	Speed East
30	0

Vectors

- The well in **Colorado** was **8000 ft deep** and **10 ft wide** with a **100 gpm** capacity.

State	Depth	Width	Capacity
CO	8000	10	100

Vectors

- The well in **Colorado** was **8000 ft deep** and **10 ft wide** with a **100 gpm** capacity.

State	Depth	Width	Capacity
CO	8000	10	100

What do these numbers mean?

(If you came back to this data two weeks from now, would you remember?)

Vectors

- The well in **Colorado** was **8000 ft deep** and **10 ft wide** with a **100 gpm** capacity.

State_abrv	Depth_ft	Width_ft	Capacity_gpm
CO	8000	10	100

Consider including units of measure in column names.

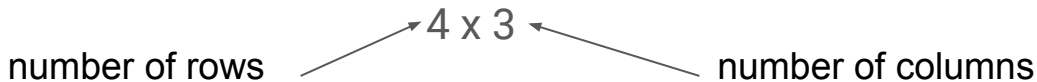
Matrices

- A **matrix** is a rectangular array of numbers arranged in rows and columns.
- Matrices are usually designated by bold, uppercase letters: **X**
- Matrices usually represent a series of observations or samples in a dataset.

well	Depth_ft	Width_ft
1	8000	10
2	1000	3
3	10000	5
4	2000	8

- The **shape** of a matrix is the number of rows and number of columns:

number of rows 4 x 3 number of columns



Matrices

- Each row or column of a matrix can be thought of as a vector

Matrix (4 x 3)

well	Depth_ft	Width_ft
1	8000	10
2	1000	3
3	10000	5
4	2000	8

Matrices

- Each row or column of a matrix can be thought of as a vector

Matrix (4 x 3)

well	Depth_ft	Width_ft
1	8000	10
2	1000	3
3	10000	5
4	2000	8



Row Vector (1 x 3)

well	Depth_ft	Width_ft
1	8000	10

Matrices

- Each row or column of a matrix can be thought of as a vector

Matrix (4 x 3)

well	Depth_ft	Width_ft
1	8000	10
2	1000	3
3	10000	5
4	2000	8

Row Vector (1 x 3)

well	Depth_ft	Width_ft
1	8000	10

Column Vector (4 x 1)

Depth_ft
8000
1000
10000
2000

Turn & Talk

By the end of this lesson, you will be able to:

- ❑ Define the terms scalar, vector, and matrix
-
1. Compare and contrast a scalar and a vector. What do they share in common? What do they not share in common?
 2. In your own words, describe what a matrix is.

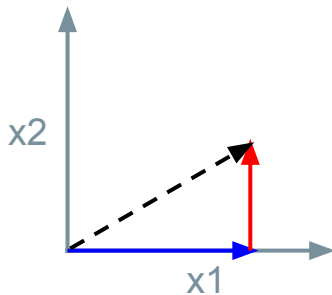
Length of a vector

Euclidean norm [\[edit \]](#)

Main article: [Euclidean distance](#)

On an n -dimensional [Euclidean space](#) \mathbf{R}^n , the intuitive notion of length of the vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$ is captured by the formula

$$\|\mathbf{x}\|_2 := \sqrt{x_1^2 + \dots + x_n^2}.$$



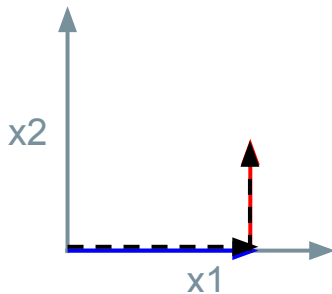
L2 norm: $\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2}$

Taxicab norm or Manhattan norm [\[edit \]](#)

Main article: [Taxicab geometry](#)

$$\|\mathbf{x}\|_1 := \sum_{i=1}^n |x_i|.$$

The name relates to the distance a taxi has to drive in a rectangular [street grid](#) to get from the origin to the point \mathbf{x} .



L1 norm: $\|\mathbf{x}\|_1 = |x_1| + |x_2|$

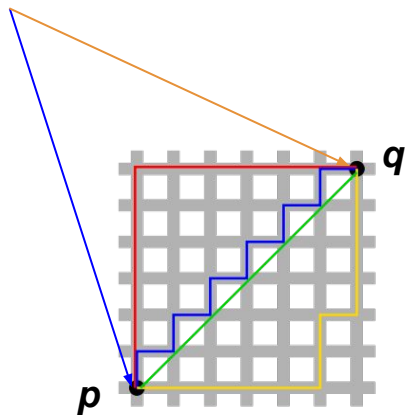
Distance between vectors

Euclidean Distance

The **Euclidean distance** between points \mathbf{p} and \mathbf{q} is the length of the **line segment** connecting them ($\overline{\mathbf{pq}}$).

In **Cartesian coordinates**, if $\mathbf{p} = (p_1, p_2, \dots, p_n)$ and $\mathbf{q} = (q_1, q_2, \dots, q_n)$ are two points in **Euclidean n -space**, then the distance (d) from \mathbf{p} to \mathbf{q} , or from \mathbf{q} to \mathbf{p} is given by the **Pythagorean formula**:^[1]

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2}$$
$$= \sqrt{\sum_{i=1}^n (q_i - p_i)^2}.$$



— euclidean

Distance between vectors

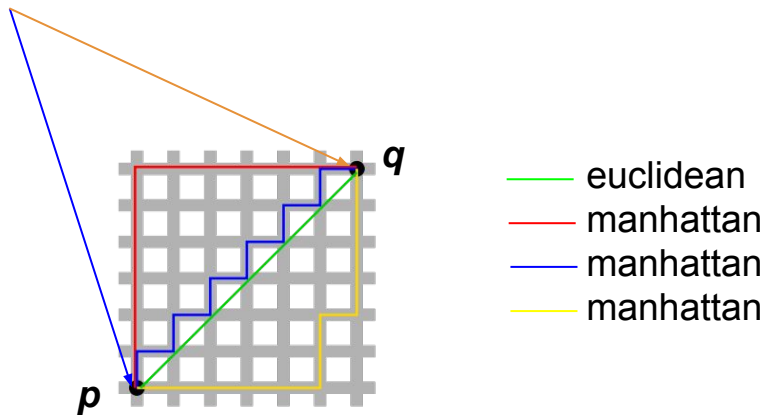
Manhattan or Taxicab Distance

The taxicab distance, d_1 , between two vectors \mathbf{p} , \mathbf{q} in an n -dimensional [real vector space](#) with fixed [Cartesian coordinate system](#), is the sum of the lengths of the projections of the [line segment](#) between the points onto the [coordinate axes](#). More formally,

$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^n |p_i - q_i|,$$

where (\mathbf{p}, \mathbf{q}) are [vectors](#)

$$\mathbf{p} = (p_1, p_2, \dots, p_n) \text{ and } \mathbf{q} = (q_1, q_2, \dots, q_n)$$



Distance between vectors

Cosine Similarity

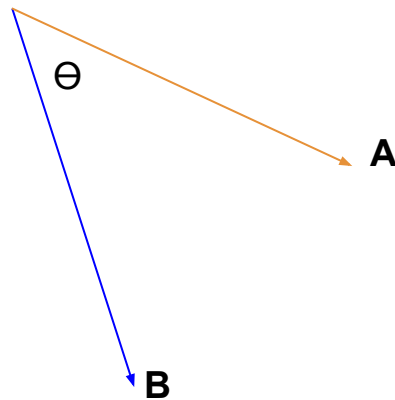
The cosine of two non-zero vectors can be derived by using the [Euclidean dot product](#) formula:

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \|\mathbf{B}\| \cos \theta$$

Given two [vectors](#) of attributes, A and B , the cosine similarity, $\cos(\theta)$, is represented using a [dot product](#) and [magnitude](#) as

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

where A_i and B_i are [components](#) of vector A and B respectively.

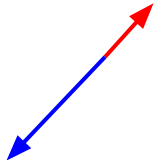
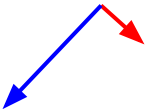
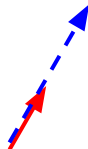


Distance between vectors

Cosine Distance

When two vectors point in the same direction, we want their distance to be small.

$$\text{cosine distance} = 1 - \text{cosine similarity}$$

Metric	Vectors pointing in opposite directions	Vectors orthogonal	Vectors pointing in the same direction
Image			
Cosine similarity	-1	0	1
Cosine distance	2	1	0

Typically used in higher dimensional vector spaces (easier to be similar)

Turn & Talk

By the end of this lesson, you will be able to:

- ❏ Compare and contrast operations that calculate length and distance of vectors
-
1. What is the difference between Euclidean distance and Manhattan distance? Which is L1 and which is L2?
 2. Why would a data scientist prefer to utilize cosine distance instead of cosine similarity?

Vector and Matrix Operations

Transpose

In [linear algebra](#), the **transpose** of a [matrix](#) is an operator which flips a matrix over its diagonal, that is it switches the row and column indices of the matrix by producing another matrix denoted as \mathbf{A}^T (also written \mathbf{A}' , \mathbf{A}^{tr} , ${}^t\mathbf{A}$ or \mathbf{A}^t).

$$\begin{bmatrix} 1 & 2 \end{bmatrix}^T = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

Vector and Matrix Operations

Dot Product

In [mathematics](#), the **dot product** or **scalar product**^[note 1] is an [algebraic operation](#) that takes two equal-length sequences of numbers (usually [coordinate vectors](#)) and returns a single number.

The dot product of two vectors $\mathbf{a} = [a_1, a_2, \dots, a_n]$ and $\mathbf{b} = [b_1, b_2, \dots, b_n]$ is defined as:^[1]

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$$

where Σ denotes [summation](#) and n is the dimension of the [vector space](#). For instance, in [three-dimensional space](#), the dot product of vectors $[1, 3, -5]$ and $[4, -2, -1]$ is:

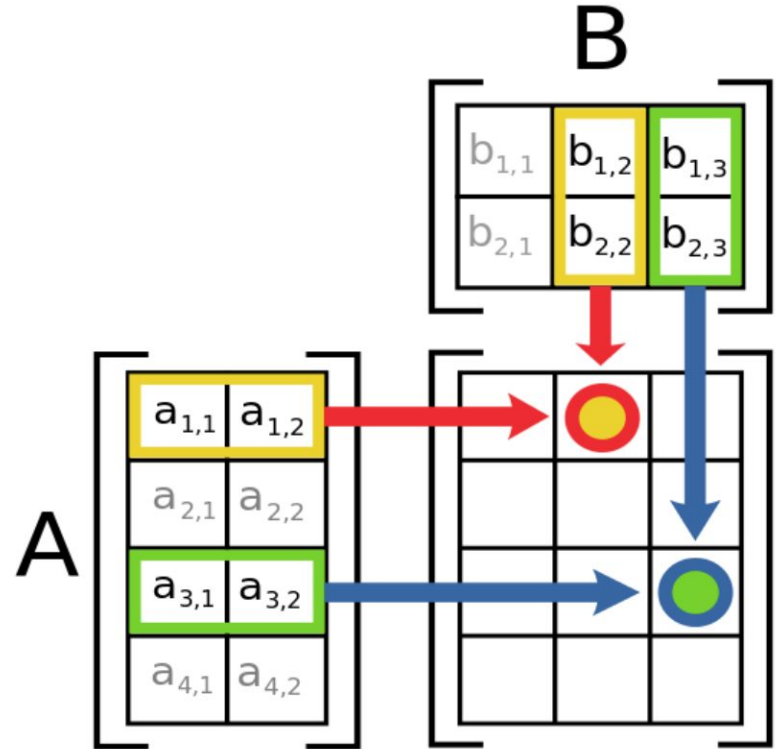
$$\begin{aligned} [1, 3, -5] \cdot [4, -2, -1] &= (1 \times 4) + (3 \times -2) + (-5 \times -1) \\ &= 4 - 6 + 5 \\ &= 3 \end{aligned}$$

Vector and Matrix Operations

Matrix Multiplication

The figure to the right illustrates diagrammatically the product of two matrices **A** and **B**, showing how each intersection in the product matrix corresponds to a row of **A** and a column of **B**.

$$\begin{array}{c} 4 \times 2 \text{ matrix} \\ \begin{bmatrix} a_{11} & a_{12} \\ \cdot & \cdot \\ a_{31} & a_{32} \\ \cdot & \cdot \end{bmatrix} \end{array} \begin{array}{c} 2 \times 3 \text{ matrix} \\ \begin{bmatrix} \cdot & b_{12} & b_{13} \\ \cdot & b_{22} & b_{23} \end{bmatrix} \end{array} = \begin{array}{c} 4 \times 3 \text{ matrix} \\ \begin{bmatrix} \cdot & x_{12} & x_{13} \\ \cdot & \cdot & \cdot \\ \cdot & x_{32} & x_{33} \\ \cdot & \cdot & \cdot \end{bmatrix} \end{array}$$



source: Wikipedia

Vector and Matrix Operations

Limitations of Matrix Multiplication

- Matrix multiplication is NOT commutative: $\mathbf{A} \bullet \mathbf{B} \neq \mathbf{B} \bullet \mathbf{A}$
- Matrix multiplication can only happen when the number of columns in the first matrix matches the number of rows in the second matrix.

$$\begin{array}{c} 4 \times \textcircled{2} \text{ matrix} \\ \begin{bmatrix} a_{11} & a_{12} \\ \cdot & \cdot \\ a_{31} & a_{32} \\ \cdot & \cdot \end{bmatrix} \end{array} \begin{array}{c} \textcircled{2} \times 3 \text{ matrix} \\ \begin{bmatrix} \cdot & b_{12} & b_{13} \\ \cdot & b_{22} & b_{23} \end{bmatrix} \end{array} = \begin{array}{c} 4 \times 3 \text{ matrix} \\ \begin{bmatrix} \cdot & x_{12} & x_{13} \\ \cdot & \cdot & \cdot \\ \cdot & x_{32} & x_{33} \\ \cdot & \cdot & \cdot \end{bmatrix} \end{array}$$

Breakout - pair up

By the end of this lesson, you will be able to:

- ❑ Perform vector dot product multiplication and and matrix multiplication

1. If $\mathbf{x} = [3, 5, 8]$ and $\mathbf{y} = [-2, -4, -1]$, what is their dot product?

For the next four problems, refer to matrices \mathbf{A} and \mathbf{B} on the right.

2. What is $\mathbf{A} \bullet \mathbf{B}$?

3. What is $\mathbf{B}^T \bullet \mathbf{A}$?

4. Which of these cannot be done? Explain why.

$$\mathbf{B} \bullet \mathbf{B}$$

$$\mathbf{B} \bullet \mathbf{B}^T$$

$$\mathbf{A} = \begin{array}{|c|c|} \hline 1 & 2 \\ \hline -1 & 0 \\ \hline \end{array}$$

$$\mathbf{B} = \begin{array}{|c|c|c|} \hline 1 & 2 & 0 \\ \hline 1 & -1 & 3 \\ \hline \end{array}$$

Vectors in Data Science

In data science, we broaden the idea of a vector to include non-numerical information.

- I bought **2 boxes of oatmeal**, **1 box of crackers**, at the **supermarket** for only **\$2.50**.

Oatmeal	Crackers	Location	Price
2	1	supermarket	2.5

Vectors in Data Science

In data science, we broaden the idea of a vector to include non-numerical information.

- I bought **2 boxes of oatmeal**, **1 box of crackers**, at the **supermarket** for only **\$2.50**.

Oatmeal	Crackers	Location	Price
2	1	supermarket	2.5



It's a categorical variable, but
models can't train on text.
How can we make this a number?

Vectors in Data Science

One can use **one-hot encoding** to transform categorical variables into numerical values.

Let's say our options (categories) are: supermarket, corner store, and on-line.

Before one-hot encoding:

Oatmeal	Crackers	Location	Price
2	1	supermarket	2.5

Vectors in Data Science

One can use **one-hot encoding** to transform categorical variables into numerical values.

Let's say our options (categories) are: supermarket, corner store, and on-line.

Before one-hot encoding:

Oatmeal	Crackers	Location	Price
2	1	supermarket	2.5

After one-hot encoding:

Oatmeal	Crackers	supermarket	corner_store	on-line	Price
2	1	1	0	0	2.5



The purchase location has been
one-hot encoded.

Vectors in Data Science

- I bought **2 boxes of oatmeal**, **1 box of crackers**, at the **supermarket** for only **\$2.50**.

Oatmeal	Crackers	supermarket	Price
2	1	1	2.5



Then again, maybe it only matters if the purchased happened at the supermarket or not...

Vectors in Data Science

Bag of Words

- Can you make a vector of this [data-science haiku](#)?

*My profile is me
A bunch of features - that's all
But where is my soul?*

Vectors in Data Science

Bag of Words

- Can you make a vector of this [data-science haiku](#)?

*My profile is me
A bunch of features - that's all
But where is my soul?*

Consider using a [bag-of-words approach](#), where you count the number of times each word occurs (term frequency).

Vectors in Data Science

Bag of Words

- Can you make a vector of this [data-science haiku](#)?

*My profile is me
A bunch of features - that's all
But where is my soul?*

Consider using a [bag-of-words approach](#), where you count the number of times each word occurs (term frequency).

a	all	bunch	but	features	is	me	my	profile	soul	that's	where



the vocabulary

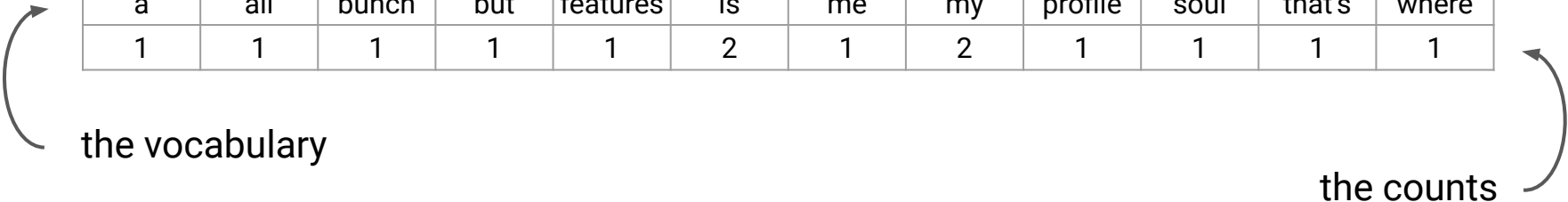
Vectors in Data Science

Bag of Words

- Can you make a vector of this [data-science haiku](#)?

*My profile is me
A bunch of features - that's all
But where is my soul?*

Consider using a [bag-of-words approach](#), where you count the number of times each word occurs (term frequency).



a	all	bunch	but	features	is	me	my	profile	soul	that's	where
1	1	1	1	1	2	1	2	1	1	1	1

the vocabulary

the counts

Vectors in Data Science

Bag of Words

If you had multiple haikus, a bag-of-words approach would result in a matrix called a term-frequency matrix.

- The term frequency matrix of these [data-science haikus](#):

My profile is me

A bunch of features - that's all

But where is my soul?

As data unfold

Alas the model won't fit

Still the curve bends right

haiku	a	all	alas	as	bends	bunch	but	curve	data	features	is	model	my	profile	right	soul	still	that's	the	unfold	where	won't
1	1	1	0	0	0	1	1	0	0	1	2	0	2	1	0	1	0	1	0	0	1	0
2	0	0	1	1	1	0	0	1	1	0	0	1	0	0	1	0	1	0	2	1	0	1

Vectors in Data Science

Flattening a matrix

- Can you make a 1-D vector (a row) describing the status of this tic-tac-toe game?

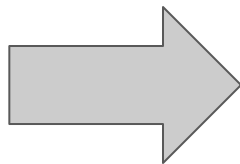
X		
	O	O
		X

Vectors in Data Science

Flattening a matrix

- Assign numerical values to each position: -1 (unplayed), 0 (O), and 1 (X).

X		
	O	O
		X



1	-1	-1
-1	0	0
-1	-1	1

Vectors in Data Science

Flattening a matrix

- Starting with the last row, append each row to the end of the row above.

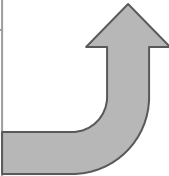
1	-1	-1
-1	0	0
-1	-1	1

Vectors in Data Science

Flattening a matrix

- Starting with the last row, append each row to the end of the row above.

1	-1	-1
-1	0	0
-1	-1	1



Vectors in Data Science

Flattening a matrix

- Starting with the last row, append each row to the end of the row above.

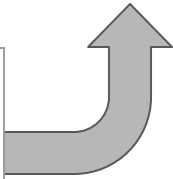
1	-1	-1			
-1	0	0	-1	-1	1

Vectors in Data Science

Flattening a matrix

- Starting with the last row, append each row to the end of the row above.

1	-1	-1			
-1	0	0	-1	-1	1



Vectors in Data Science

Flattening a matrix

- Starting with the last row, append each row to the end of the row above.

1	-1	-1	-1	0	0	-1	-1	1
---	----	----	----	---	---	----	----	---

Vectors in Data Science

Flattening a matrix

X		
	O	O
		X

1	-1	-1
-1	0	0
-1	-1	1

TL	TM	TR
ML	M	MR
BL	BM	BR

The flattened (1-D) vector:

TL	TM	TR	ML	M	MR	BL	BM	BR
1	-1	-1	-1	0	0	-1	-1	1

Vectors in Data Science

- Almost any type of data can be vectorized/featurized (doesn't always have to be 1-D).
- Almost always, there is more than one way to do it.
- Your ability to featurize and [feature engineer](#) your data will be fundamental to the success of your machine learning algorithms and your career as a data scientist.
- Your intuition will grow with experience (and by reading books, papers, and blogs!)

Turn & Talk

By the end of this lesson, you will be able to:

- ❑ Describe three techniques data scientists use to featurize real-world data

Describe the data science techniques you might use when featurizing the following data:

1. A color image
2. A song
3. A movie review

NumPy

NumPy is the fundamental package for scientific computing in Python.

It provides:

- multidimensional array object, the [ndarray](#)
 - encapsulates n -dimensional arrays of homogeneous data types stored (ideally) sequentially in memory
- routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, **basic linear algebra**, basic statistical operations, random simulation and much more.
 - mostly written in C

NumPy

There are a couple important differences between NumPy arrays and standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an *ndarray* will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.

NumPy demo

`linear-algebra-with-numpy.ipynb`

Objectives

By the end of this lesson, you will be able to:

- ❑ Define the terms scalar, vector, and matrix
- ❑ Compare and contrast operations that calculate length and distance of vectors
- ❑ Perform vector dot product multiplication and matrix multiplication
- ❑ Describe three techniques data scientists use to featurize real-world data
- ❑ Use NumPy to perform linear algebra on scalars, vectors, and matrices