Big Data Computing

Master's Degree in Computer Science

2024-2025

Gabriele Tolomei

Department of Computer Science

Sapienza Università di Roma

tolomei@di.uniroma1.it

SAPIENZA UNIVERSITÀ DI ROMA

Who Am I?

Who Am I?

UniPI
(1999-2005)

Who Am I?

UniPI
(1999-2005)

UniVE
(2008-2013)

Who Am I?

UniPI
(1999-2005)

UniVE
(2008-2013)

Yahoo! Labs
(2014-2017)

Who Am I?

UniPI
(1999-2005)

UniVE
(2008-2013)

Yahoo! Labs
(2014-2017)

UniPD
(2017-2019)

Who Am I?

UniPI
(1999-2005)

UniVE
(2008-2013)

Yahoo! Labs
(2014-2017)

UniPD
(2017-2019)

Sapienza
(2019-)

My Research Interests

HERCOLE Lab

My Research Interests

Human-Explainable

HERCOLE Lab

My Research Interests

Robust

Human-Explainable

HERCOLE Lab

My Research Interests

Robust

Human-Explainable

COllaborative

HERCOLE Lab

My Research Interests

Robust

Human-Explainable

COllaborative

HERCOLE Lab

LEarning

My Research Interests

Sounds cool?

HERCOLE Lab

My Research Interests

Sounds cool?

HERCOLE Lab

Check out the lab's home page

(still under construction, sic!)

My Research Interests

Sounds cool?

HERCOLE Lab

Meanwhile you can follow us on Twitter @HercoleLab

Administrivia

- Class schedule:
  - **Wednesday** from **10:00 a.m.** to **12:00 p.m.**

Aula Magna
Viale Regina Elena, 295

Administrivia

- Class schedule:
  - **Wednesday** from **10:00 a.m.** to **12:00 p.m.**
  - **Thursday** from **1:00 p.m.** to **4:00 p.m.**

Aula Magna
Viale Regina Elena, 295

Aula 1L
Via Castro Laurenziano, 7

Administrivia

- Class schedule:
  - **Wednesday** from **10:00 a.m.** to **12:00 p.m.**
  - **Thursday** from **1:00 p.m.** to **4:00 p.m.**

Aula Magna
Viale Regina Elena, 295

Aula 1L
Via Castro Laurenziano, 7

- Office hours:
  - Drop me a message to ask for a meeting **online** (Google Meet or Zoom) or in-person at my office (Room 106 @ Viale Regina Elena, 295 – 1st Floor, Building E)

Administrivia

- Contacts:
    - Personal homepage: https://www.di.uniroma1.it/~tolomei
    - Email: tolomei@di.uniroma1.it

Administrivia

- Resources:
  - Course's website: https://github.com/gtolomei/big-data-computing
  - Moodle's web page: https://elearning.uniroma1.it/course/view.php?id=18525

Administrivia

- Resources:
  - Course's website: https://github.com/gtolomei/big-data-computing
  - Moodle's web page: https://elearning.uniroma1.it/course/view.php?id=18525
- Class material will be published on the course's website **only**
  - Along with other resources (e.g., suggested readings/books) if needed

Administrivia

- Prerequisites:
  - Familiarity with basics of Data Science and Machine Learning
  - Solid knowledge of Calculus, Linear Algebra, and Probability&Statistics
  - (Python) Programming skills desirable yet not mandatory!

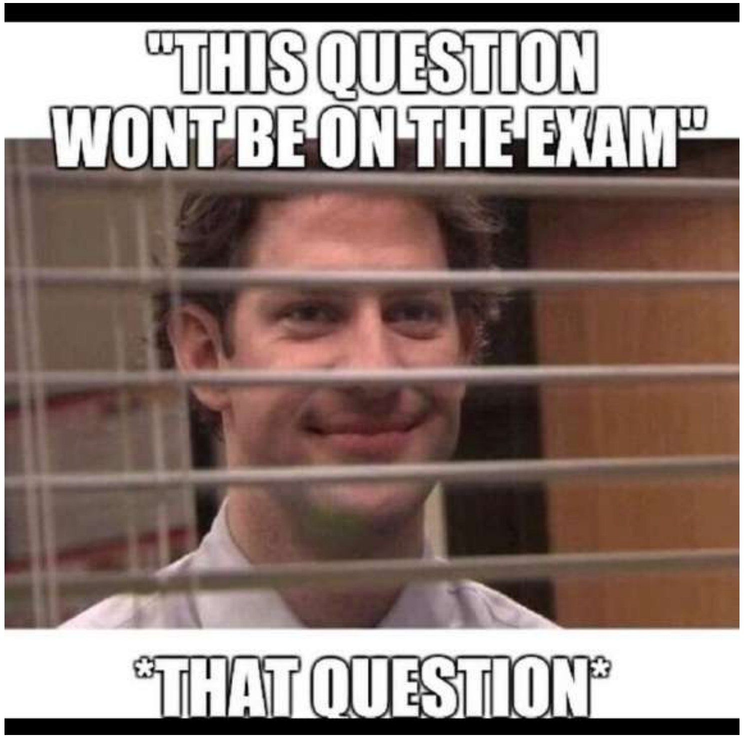
Administrivia

- Prerequisites:
  - Familiarity with basics of Data Science and Machine Learning
  - Solid knowledge of Calculus, Linear Algebra, and Probability&Statistics
  - (Python) Programming skills desirable yet not mandatory!

No worries!
Many subjects will be anyway revisited during class lectures

Administrivia

- Exam: Oral questions on the *whole* program of the course!

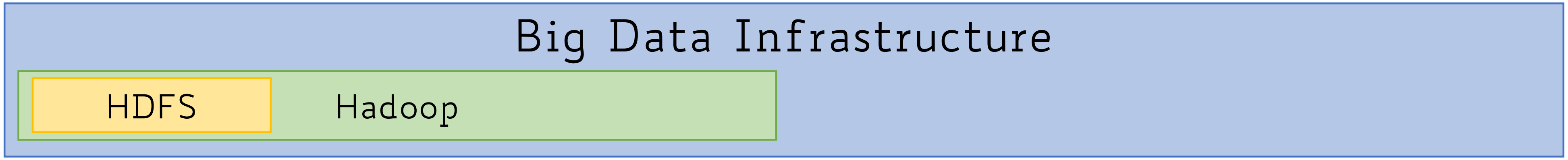


Questions?

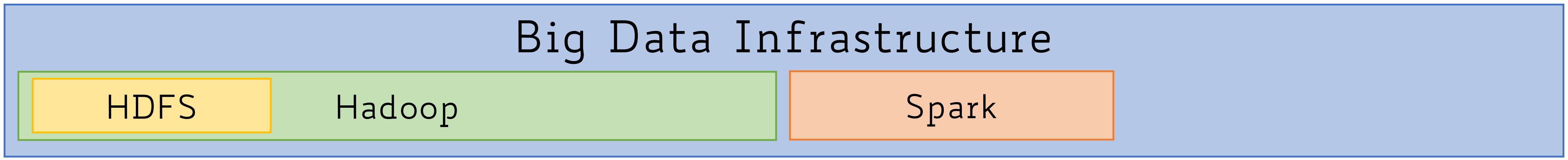
Outline of the Course

Big Data Infrastructure

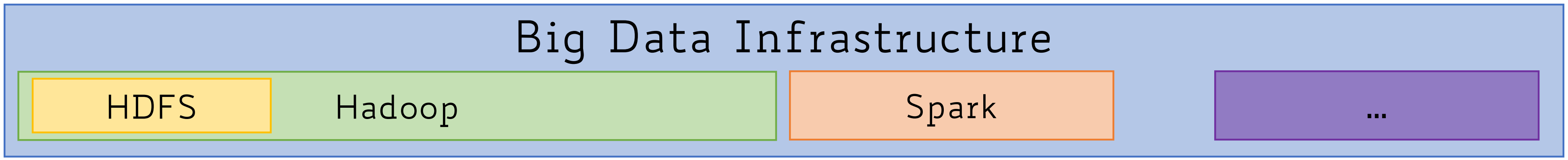
Outline of the Course



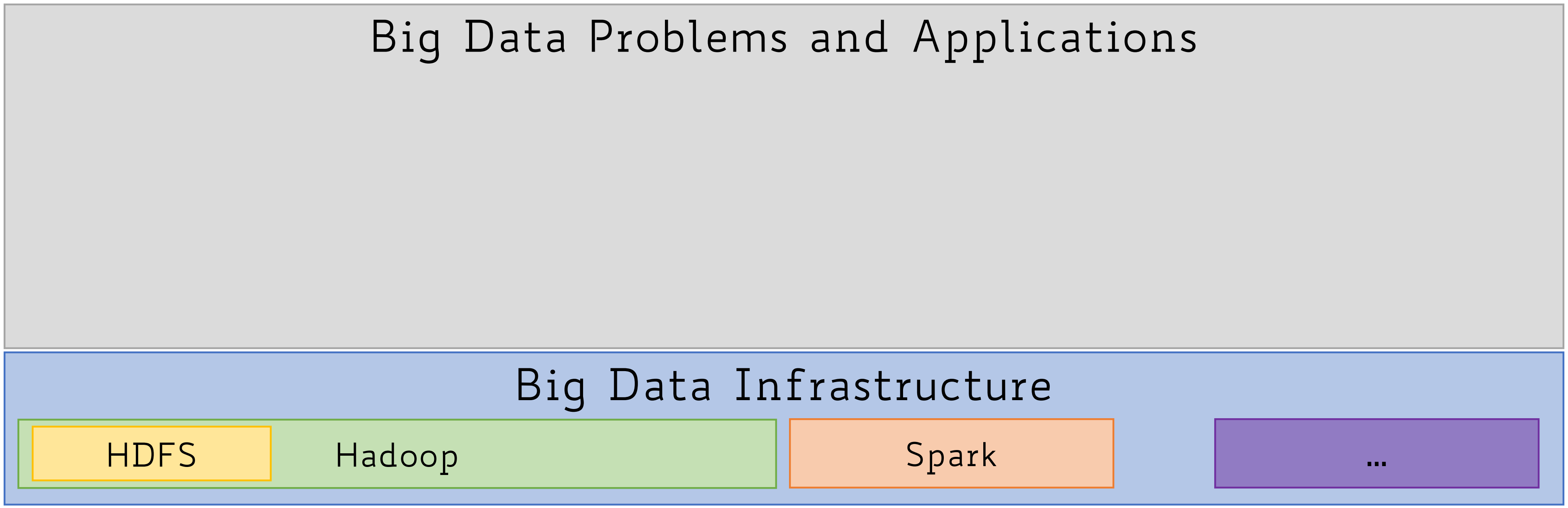
Outline of the Course



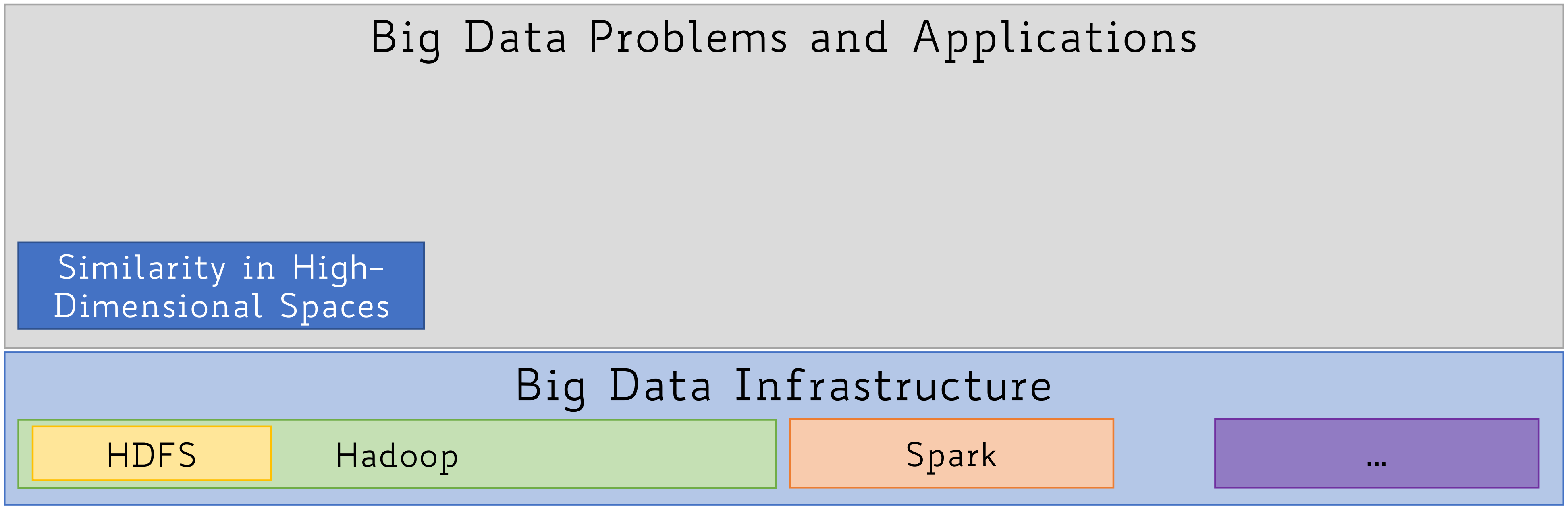
Outline of the Course



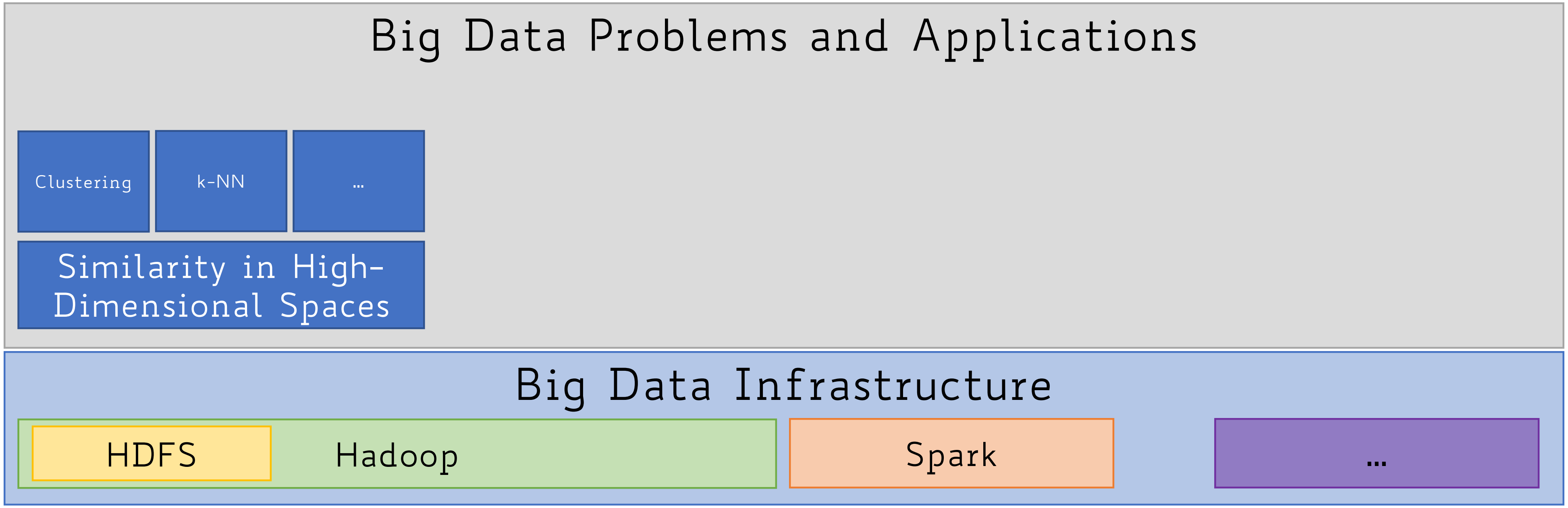
Outline of the Course



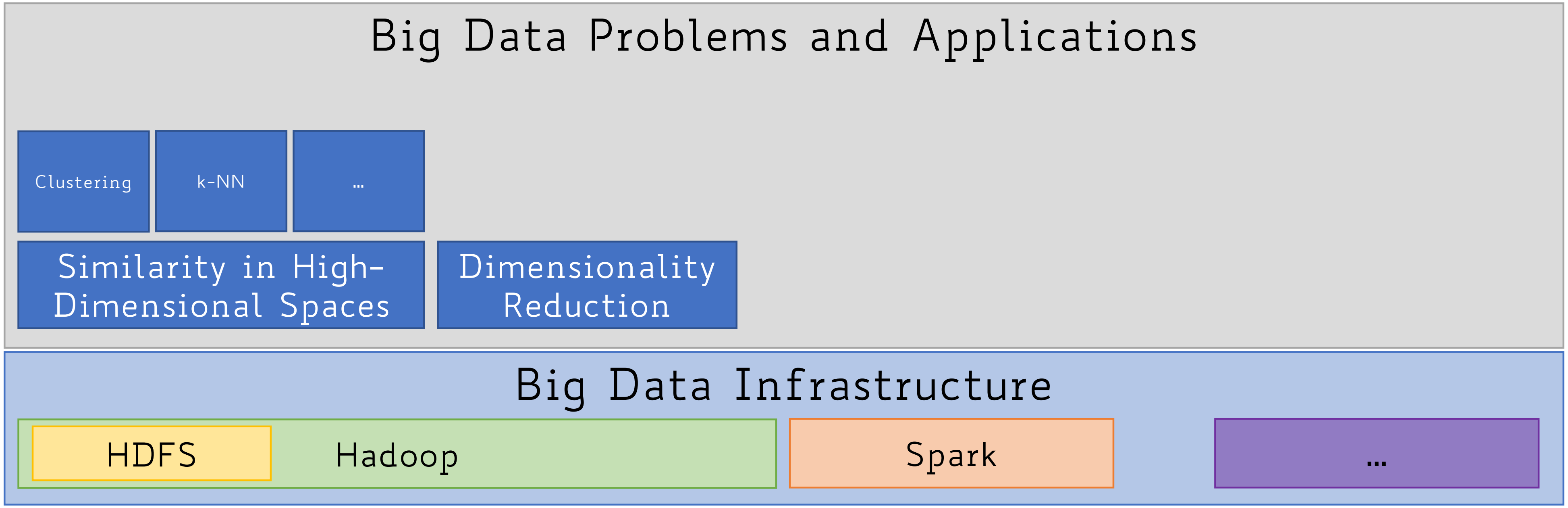
Outline of the Course



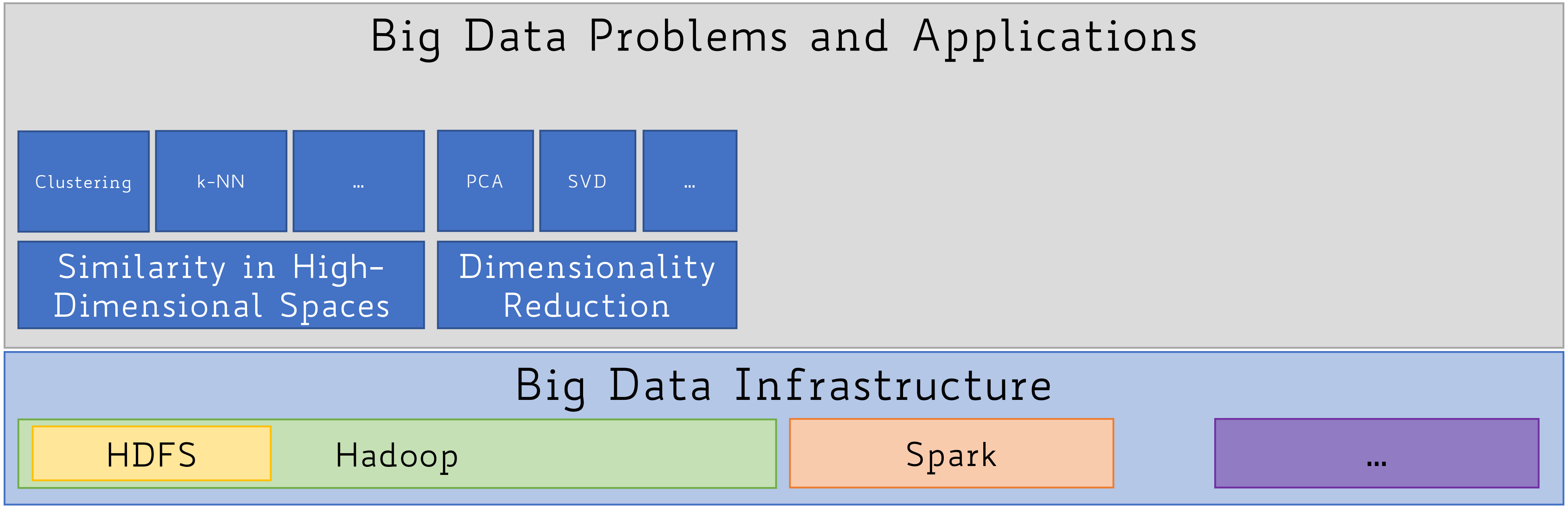
Outline of the Course



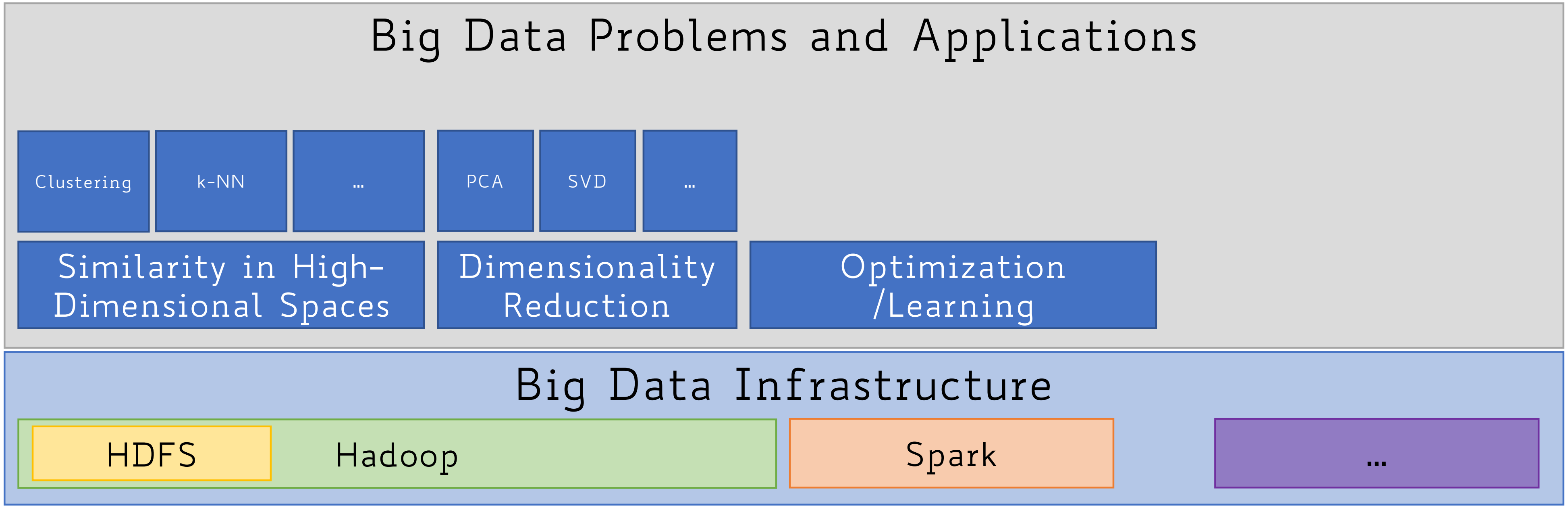
Outline of the Course



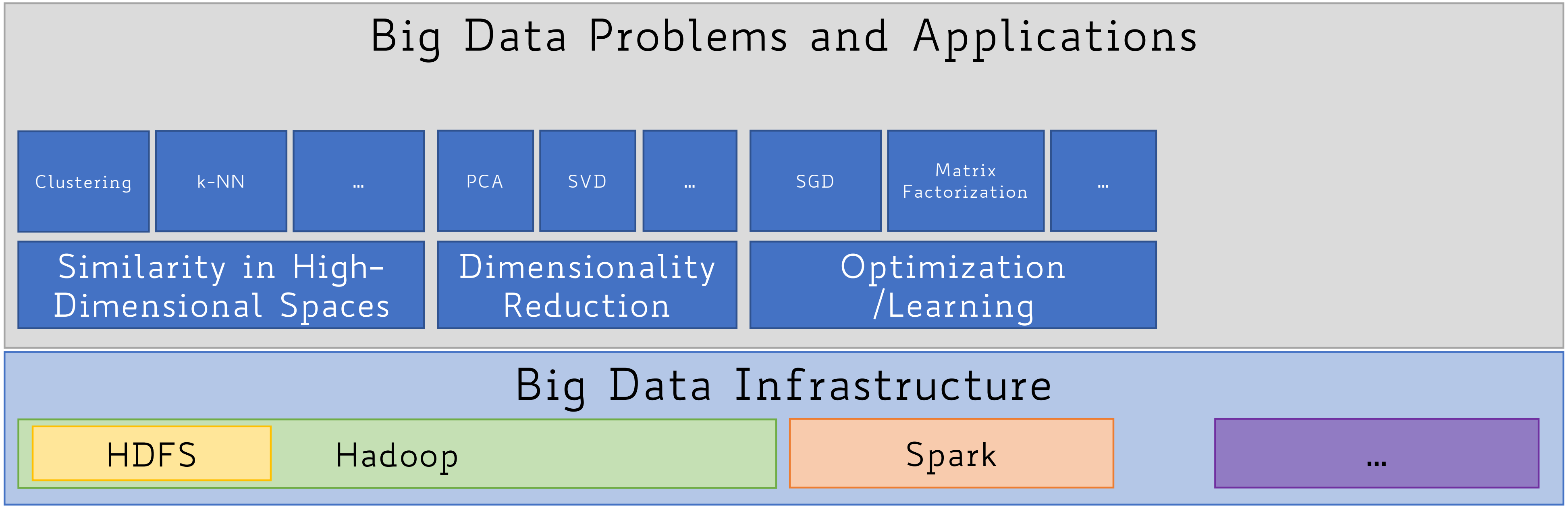
Outline of the Course



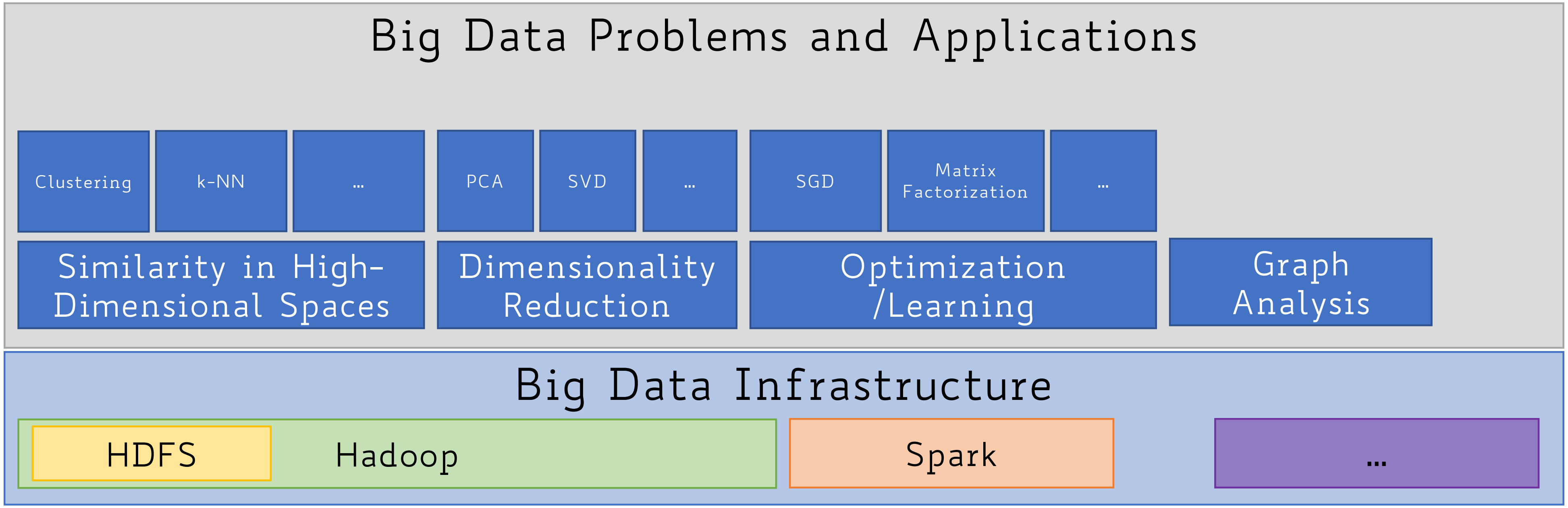
Outline of the Course



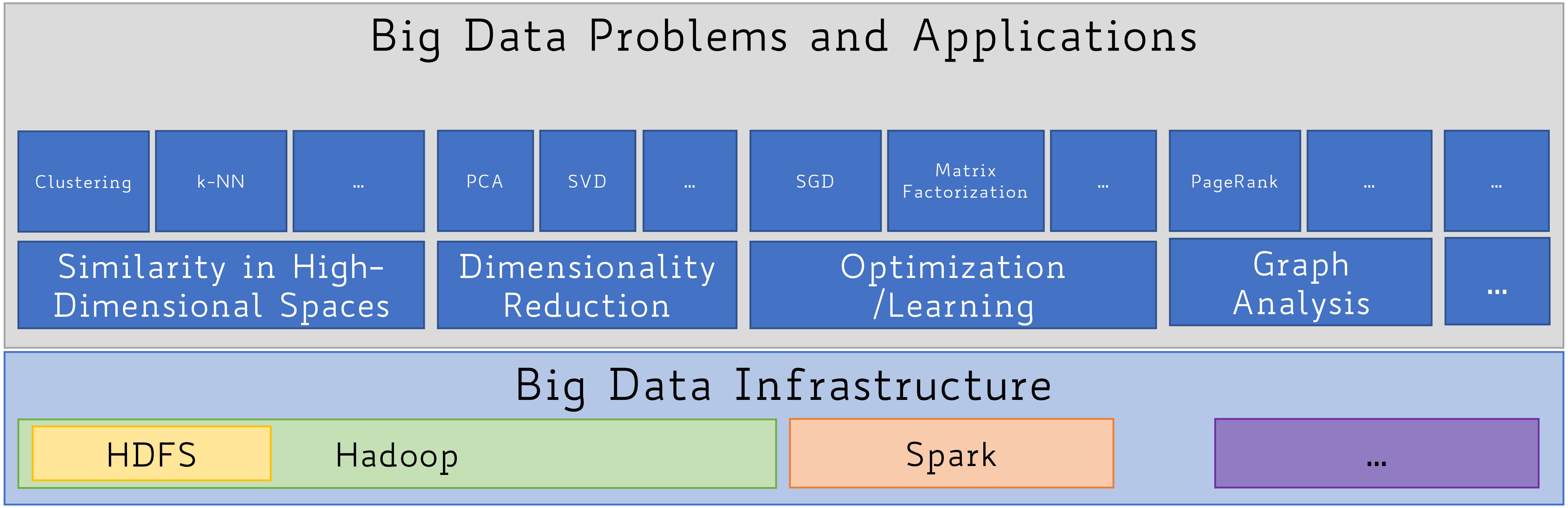
Outline of the Course



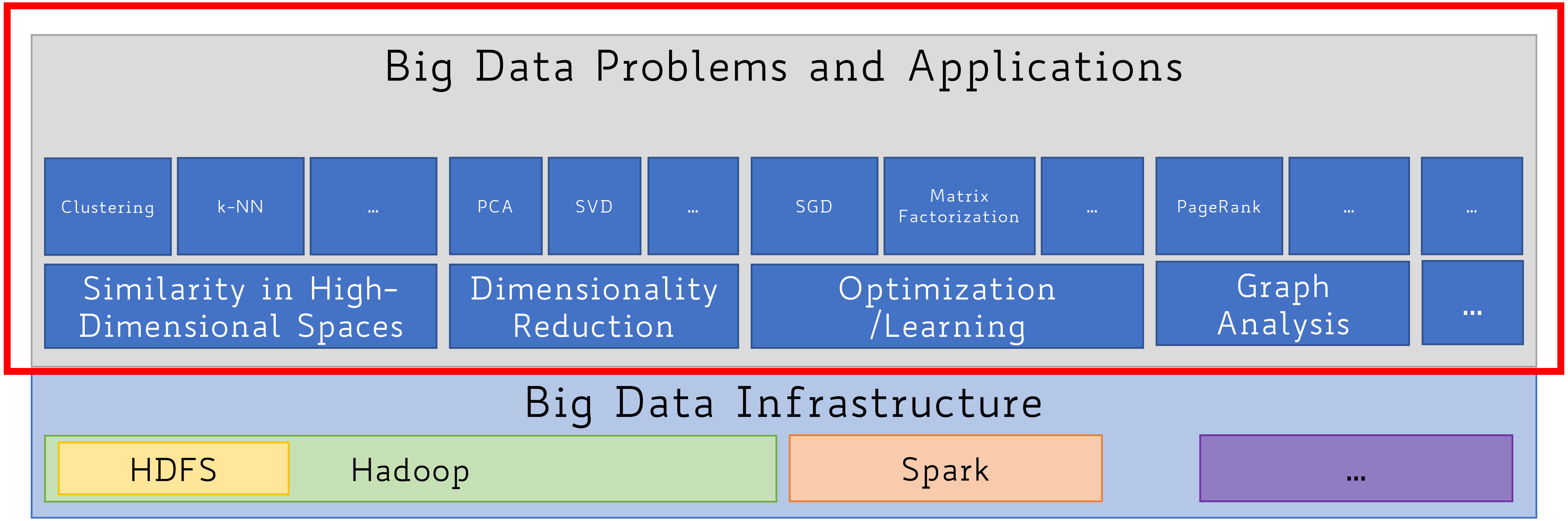
Outline of the Course



Outline of the Course



Outline of the Course



Let's Get Started!

The Search Problem

Given a collection S of N items, find if x ∈ S

The Search Problem

Given a collection S of N items, find if $x \in S$

This is a very general problem that occurs frequently

The Search Problem

Given a collection S of N items, find if x ∈ S

This is a very general problem that occurs frequently

Example:

Find if a number occurs in a list of N integers

The Search Problem

If the list is not sorted, it takes O(N) steps to respond

The Search Problem

If the list is not sorted, it takes O(N) steps to respond

If the list is sorted, it takes O(log(N)) steps to respond but we must pay the cost of sorting it O(Nlog(N))

The Search Problem

If the list is not sorted, it takes O(N) steps to respond

If the list is sorted, it takes O(log(N)) steps to respond but we must pay the cost of sorting it O(Nlog(N))

Still, it might be beneficial to pre-sort the list if we repeat the find operation several times...

Time Complexity

Let M be the number of times the find operation is called

Time Complexity

Let M be the number of times the find operation is called

Without sorting, the total complexity is O(M*N) steps

Time Complexity

Let M be the number of times the find operation is called

Without sorting, the total complexity is $O(M*N)$ steps

With sorting, the complexity becomes $O(N\log(N)) + O(M\log(N))$

Time Complexity

Let M be the number of times the find operation is called

Without sorting, the total complexity is O(M*N) steps

With sorting, the complexity becomes O(Nlog(N)) + O(Mlog(N))

We must find the value of M for which:

$$\underbrace{O(N\log(N)) + O(M\log(N))}_{\text{with sorting}} < \underbrace{O(M*N)}_{\text{without sorting}}$$

Time Complexity Trade-Off

Brutally:

$$N\log(N) + M\log(N) < MN$$

Time Complexity Trade-Off

Brutally:

$$N\log(N) + M\log(N) < MN$$

$$M\log(N) - MN < -N\log(N)$$

Time Complexity Trade-Off

Brutally:

$$N\log(N) + M\log(N) < MN$$

$$M\log(N) - MN < -N\log(N)$$

$$MN - M\log(N) > N\log(N)$$

Time Complexity Trade-Off

Brutally:

$$N\log(N) + M\log(N) < MN$$

$$M\log(N) - MN < -N\log(N)$$

$$MN - M\log(N) > N\log(N)$$

$$M(N-\log(N)) > N\log(N)$$

Time Complexity Trade-Off

Brutally:

$$N\log(N) + M\log(N) < MN$$

$$M\log(N) - MN < -N\log(N)$$

$$MN - M\log(N) > N\log(N)$$

$$M(N-\log(N)) > N\log(N)$$

$$\boxed{M >= \lceil N\log(N)/(N-\log(N))\rceil \text{ [assuming } N \in \mathbb{Z}_{>0}\text{]}}$$

Time Complexity Trade-Off: Example

N	M	M*N	NlogN + MlogN

Time Complexity Trade-Off: Example

N	M	M*N	NlogN + MlogN
1000			

Time Complexity Trade-Off: Example

N	M	M*N	NlogN + MlogN
1000	2		

Time Complexity Trade-Off: Example

N	M	M*N	NlogN + MlogN
1000	2	2000	

Time Complexity Trade-Off: Example

N	M	M*N	NlogN + MlogN
1000	2	2000	1000*3 + 2*3 = 3006

Time Complexity Trade-Off: Example

N	M	M*N	NlogN + MlogN
1000	2	2000	1000*3 + 2*3 = 3006

Time Complexity Trade-Off: Example

N	M	M*N	NlogN + MlogN
1000	2	2000	1000*3 + 2*3 = 3006
1000	3	3000	1000*3 + 3*3 = 3009
1000	4	4000	1000*3 + 4*3 = 3012

Time Complexity Trade-Off: Example

N	M	M*N	NlogN + MlogN
1000	2	2000	1000*3 + 2*3 = 3006
1000	3	3000	1000*3 + 3*3 = 3009
1000	4	4000	1000*3 + 4*3 = 3012

$$M >= \lceil N\log(N)/(N-\log(N)) \rceil$$ [assuming $N \in \mathbb{Z}_{>0}$]

$$M >= \lceil 1000*\log(1000)/(1000-\log(1000)) \rceil$$

$$M >= \lceil 3000/(1000-3) \rceil$$

$$M >= \lceil 3.009 \rceil = 4$$

‘Vertical’ vs. ‘Horizontal’ Scale

What could make the search problem hard?

‘Vertical’ vs. ‘Horizontal’ Scale

What could make the search problem hard?

If N grows large and the list is not sorted, a linear scan can be too costly

'Vertical' vs. 'Horizontal' Scale

What could make the search problem hard?

If N grows large and the list is not sorted, a linear scan can be too costly

On the other hand, if N does not fit into main memory, we will need some external R-way merge-sorting

‘Vertical’ vs. ‘Horizontal’ Scale

What could make the search problem hard?

If N grows large and the list is not sorted, a linear scan can be too costly

On the other hand, if N does not fit into main memory, we will need some external R-way merge-sorting

In both cases, the complexity may arise from the fact that we are dealing with a massive ‘vertical’ input size N

‘Vertical’ vs. ‘Horizontal’ Scale

What could make the search problem hard?

If N grows large and the list is not sorted, a linear scan can be too costly

On the other hand, if N does not fit into main memory, we will need some external R-way merge-sorting

In both cases, the complexity may arise from the fact that we are dealing with a massive ‘vertical’ input size N

‘vertical’ here means that the number N of input data points is large, yet their representation (e.g., 8-byte integers) is not!

‘Vertical’ vs. ‘Horizontal’ Scale

Suppose that S is a collection of images, not integers

‘Vertical’ vs. ‘Horizontal’ Scale

Suppose that S is a collection of images, not integers

Our goal is to find if a given image x is in S
(or retrieve an image x’ in S that is `most similar’ to x)

‘Vertical’ vs. ‘Horizontal’ Scale

Suppose that S is a collection of images, not integers

Our goal is to find if a given image x is in S
(or retrieve an image x′ in S that is `most similar’ to x)

We may have to cope both with the large number of items N in the collection and the complex representation of each item

‘Vertical’ vs. ‘Horizontal’ Scale

Suppose that S is a collection of images, not integers

Our goal is to find if a given image x is in S
(or retrieve an image x’ in S that is `most similar’ to x)

We may have to cope both with the large number of items N in the collection and the complex representation of each item

Each image must be represented by a high-dimensional vector of RGB pixels

‘Vertical’ vs. ‘Horizontal’ Scale

Suppose that S is a collection of images, not integers

Our goal is to find if a given image x is in S
(or retrieve an image x′ in S that is `most similar’ to x)

We may have to cope both with the large number of items N in the collection and the complex representation of each item

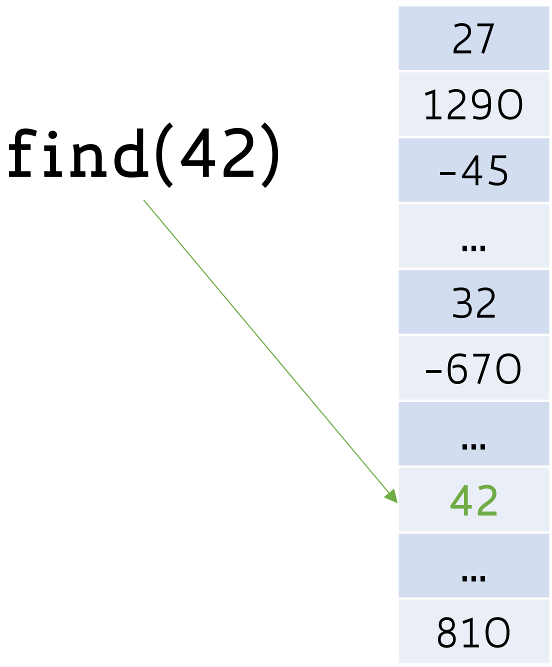
Each image must be represented by a high-dimensional vector of RGB pixels

For example, a 100x100 pixel image requires a 30000-dimensional real-value vector

‘Vertical’ vs. ‘Horizontal’ Scale

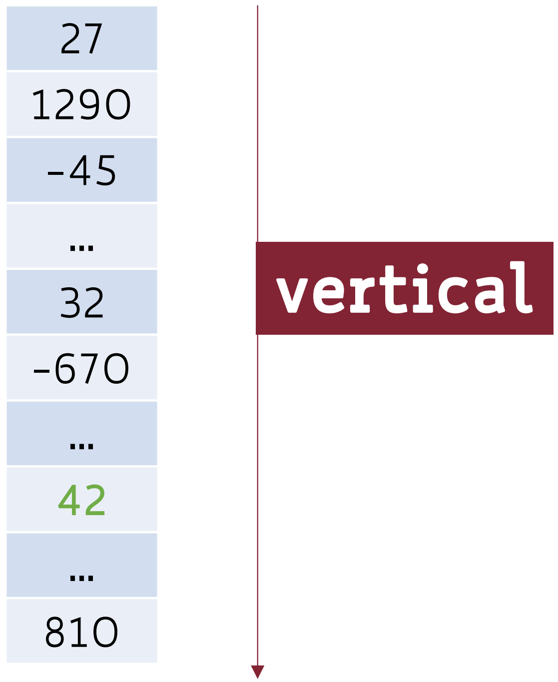
27
1290
-45
…
32
-670
…
42
…
810

‘Vertical’ vs. ‘Horizontal’ Scale



‘Vertical’ vs. ‘Horizontal’ Scale

`find(42)`

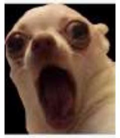


‘Vertical’ vs. ‘Horizontal’ Scale

`find(42)`

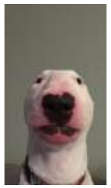
27
1290
-45
…
32
-670
…
42
…
810

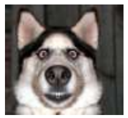
vertical



…

…



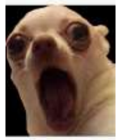


‘Vertical’ vs. ‘Horizontal’ Scale

find(42)

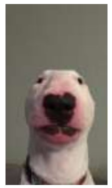
27
1290
-45
…
32
-670
…
42
…
810

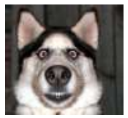
vertical



…

…





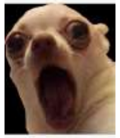
find()

‘Vertical’ vs. ‘Horizontal’ Scale

find(42)

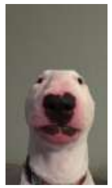
27
1290
-45
…
32
-670
…
42
…
810

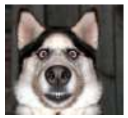
vertical



…

…





find()

horizontal

‘Vertical’ vs. ‘Horizontal’ Scale

We can experience both types of scalability issues

‘Vertical’ vs. ‘Horizontal’ Scale

We can experience both types of scalability issues

Very large number of input data points
(vertical)

AND

Very large representation of input data points
(horizontal)

‘Vertical’ vs. ‘Horizontal’ Scale

We can experience both types of scalability issues

Very large number of input data points
(vertical)

AND

Very large representation of input data points
(horizontal)

We will focus mostly on the second and still assume we deal with very large number of input data points

Similarity Measures

- What does "**similar**" mean?

Similarity Measures

- What does "**similar**" mean?
- No single answer! It depends on what we want to find or emphasize in the data

Similarity Measures

- What does "**similar**" mean?
- No single answer! It depends on what we want to find or emphasize in the data
- Domain and representation specific (e.g., similar images vs. similar text documents)

Similarity Measures

- What does "**similar**" mean?
- No single answer! It depends on what we want to find or emphasize in the data
- Domain and representation specific (e.g., similar images vs. similar text documents)
- Crucial for several big data tasks (e.g., search/retrieval/filter, clustering, classification, etc.)

Notion of Similarity

- We implicitly assumed data live in a d-dimensional **Euclidean space**

Notion of Similarity

- We implicitly assumed data live in a *d*-dimensional **Euclidean space**
- Similarity between data is computed using:
  - **Euclidean metric** (i.e., distance)
  - **Cosine similarity**
  - **Jaccard coefficient**
  - …

Metric and Metric Space

X is a set
δ is a function $\delta : X \times X \rightarrow [0, \infty)$, where:

1. $\delta(x, y) \geq 0$ (**non-negativity**)
2. $\delta(x, y) = 0 \Leftrightarrow x = y$ (**identity** of indiscernibles)
3. $\delta(x, y) = \delta(y, x)$ (**symmetry**)
4. $\delta(x, y) \leq \delta(x, z) + \delta(z, y)$ (**triangle inequality**)

Then δ is called a **metric** (or distance function) and X a **metric space**

Euclidean Metric (Distance) & Euclidean Space

$$X = \mathbb{R}^d$$

$$\delta : \mathbb{R}^d \times \mathbb{R}^d \to [0, \infty)$$

$\mathbf{x} = (x_1, \ldots, x_d)$ and $\mathbf{y} = (y_1, \ldots, y_d)$ are 2 points in $\mathbb{R}^d$

$$\delta(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + \ldots + (x_d - y_d)^2} = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

Euclidean Norm (L^2-Norm)

- The position of a point in a Euclidean d-space is a **Euclidean vector**

Euclidean Norm (L^2-Norm)

- The position of a point in a Euclidean d-space is a **Euclidean vector**
- The **Euclidean norm** of a vector measures its length (from the origin)

Euclidean Norm (L^2-Norm)

- The position of a point in a Euclidean *d*-space is a **Euclidean vector**
- The **Euclidean norm** of a vector measures its length (from the origin)

$$||\mathbf{x}||_2 = \sqrt{x_1^2 + \ldots + x_d^2} = \sqrt{\mathbf{x} \cdot \mathbf{x}}$$

where $\cdot$ indicates the **dot product**

Euclidean Norm (L^2-Norm)

- The position of a point in a Euclidean *d*-space is a **Euclidean vector**
- The **Euclidean norm** of a vector measures its length (from the origin)

$$||\mathbf{x}||_2 = \sqrt{x_1^2 + \ldots + x_d^2} = \sqrt{\mathbf{x} \cdot \mathbf{x}}$$

where $\cdot$ indicates the **dot product**

This can be just seen as the Euclidean distance between vector's tail and tip

Euclidean Norm & Euclidean Metric

Let $\mathbf{x}-\mathbf{y} = (x_1-y_1, \ldots, x_d-y_d)$ the **displacement vector** between $\mathbf{x}$ and $\mathbf{y}$

Euclidean Norm & Euclidean Metric

Let $\mathbf{x} - \mathbf{y} = (x_1 - y_1, \ldots, x_d - y_d)$ the **displacement vector** between $\mathbf{x}$ and $\mathbf{y}$

The Euclidean distance between x and y is just the Euclidean norm of the displacement vector

Euclidean Norm & Euclidean Metric

Let $\mathbf{x}-\mathbf{y} = (x_1 - y_1, \ldots, x_d - y_d)$ the **displacement vector** between $\mathbf{x}$ and $\mathbf{y}$

The Euclidean distance between x and y is just the Euclidean norm of the displacement vector

$$\delta(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_2 = \sqrt{(\mathbf{x} - \mathbf{y}) \cdot (\mathbf{x} - \mathbf{y})}$$

Euclidean Distance: 1-dimensional Case

$$d = 1$$

$$\mathbf{x}, \mathbf{y} \in \mathbb{R}^d = \mathbb{R}$$

$$\mathbf{x} = x, \mathbf{y} = y \text{ both } \mathbf{x} \text{ and } \mathbf{y} \text{ are scalars}$$

$$\delta(\mathbf{x}, \mathbf{y}) = \delta(x, y) = \sqrt{(x - y)^2} = |x - y|$$

Euclidean Distance: 1-dimensional Case

$$d = 1$$

$$\mathbf{x}, \mathbf{y} \in \mathbb{R}^d = \mathbb{R}$$

$$\mathbf{x} = x, \mathbf{y} = y \text{ both } \mathbf{x} \text{ and } \mathbf{y} \text{ are scalars}$$

$$\delta(\mathbf{x}, \mathbf{y}) = \delta(x, y) = \sqrt{(x-y)^2} = |x - y|$$

The Euclidean distance between any two 1-d points on the real line is the absolute value of the numerical difference of their coordinates

Euclidean Distance: 2-dimensional Case

$$d = 2$$

$$\mathbf{x}, \mathbf{y} \in \mathbb{R}^d = \mathbb{R}^2$$

$$\mathbf{x} = (x_1, x_2), \mathbf{y} = (y_1, y_2)$$

$$\delta(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} = ||\mathbf{x} - \mathbf{y}||_2$$

Euclidean Distance: 2-dimensional Case

$$d = 2$$
$$\mathbf{x}, \mathbf{y} \in \mathbb{R}^d = \mathbb{R}^2$$
$$\mathbf{x} = (x_1, x_2), \mathbf{y} = (y_1, y_2)$$

$$\delta(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} = ||\mathbf{x} - \mathbf{y}||_2$$

The Euclidean distance between any two 2-d points on the Euclidean plane equals to the Pythagorean theorem

Minkowski Distance (L^p-Norm)

Generalization of the Euclidean distance

$$\mathbf{x} = (x_1, \ldots, x_d) \text{ and } \mathbf{y} = (y_1, \ldots, y_d) \in \mathbb{R}^d$$

$$\delta_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Minkowski Distance (L^p-Norm): p=1

L^1-Norm or Manhattan Distance

$$\delta_1(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^1 \right)^{\frac{1}{1}} = \sum_{i=1}^{d} |x_i - y_i|$$

Minkowski Distance (L^p-Norm): p=2

L^2-Norm or Euclidean Distance

$$\delta_2(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^2\right)^{\frac{1}{2}} = \sqrt{\sum_{i=1}^{d} |x_i - y_i|^2}$$

Minkowski Distance (L^p-Norm): p=∞

$\boxed{L^{\infty}\text{-Norm or Chebyshev Distance}}$

$$\delta_\infty(\mathbf{x}, \mathbf{y}) = \lim_{p \to \infty} \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{\frac{1}{p}} =$$

$$= \max\{|x_1 - y_1|, |x_2 - y_2|, \ldots, |x_d - y_d|\}$$

Cosine Similarity

- A measure of similarity between two non-zero vectors of an inner product space

Cosine Similarity

- A measure of similarity between two non-zero vectors of an inner product space
- Measures the **cosine of the angle** between vectors

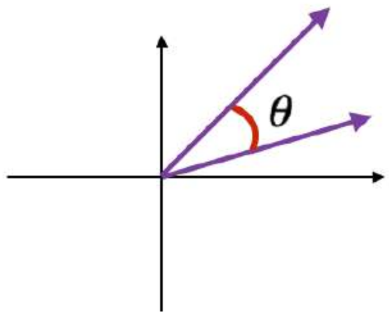
Cosine Similarity

- A measure of similarity between two non-zero vectors of an inner product space
- Measures the **cosine of the angle** between vectors
- It ranges between [-1,1]

Cosine Similarity

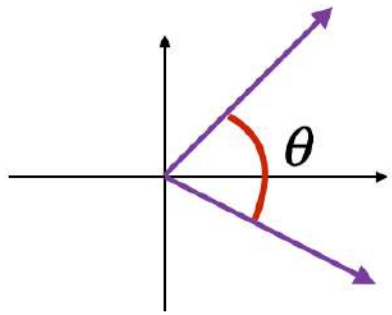
- A measure of similarity between two non-zero vectors of an inner product space
- Measures the **cosine of the angle** between vectors
- It ranges between [-1,1]
- It captures the **orientation** and not the magnitude

Cosine Similarity



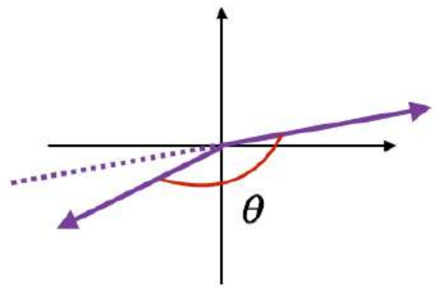
θ is close to 0°

$\cos(\theta) \approx 1$



θ is close to 90°

$\cos(\theta) \approx 0$



θ is close to 180°

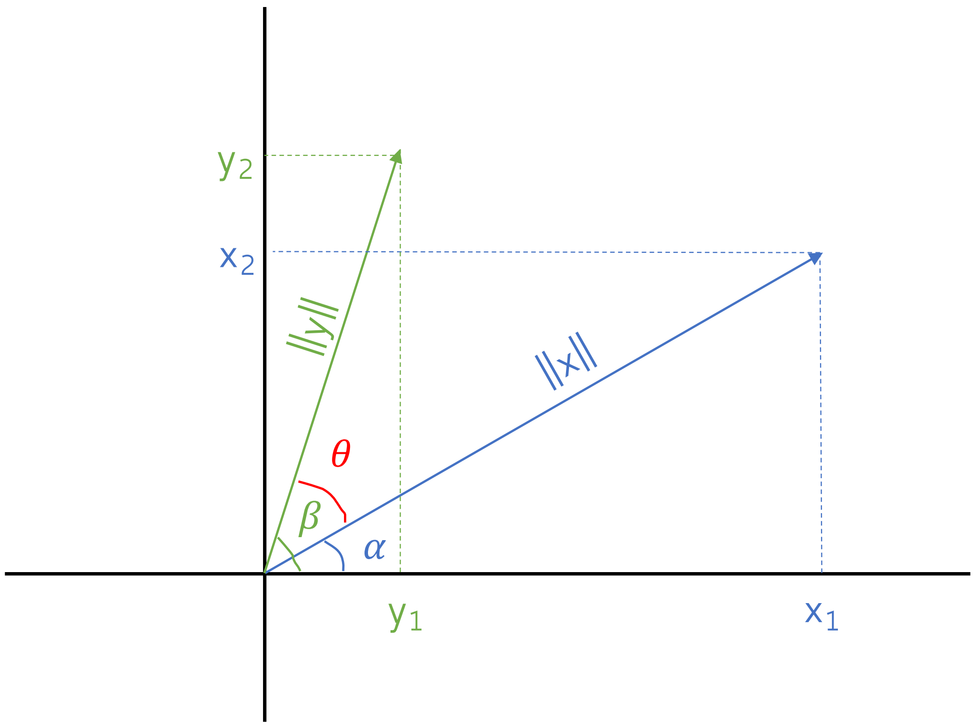
$\cos(\theta) \approx -1$

similar vectors

orthogonal vectors

opposite vectors

Cosine Similarity: 2-dimensional Case

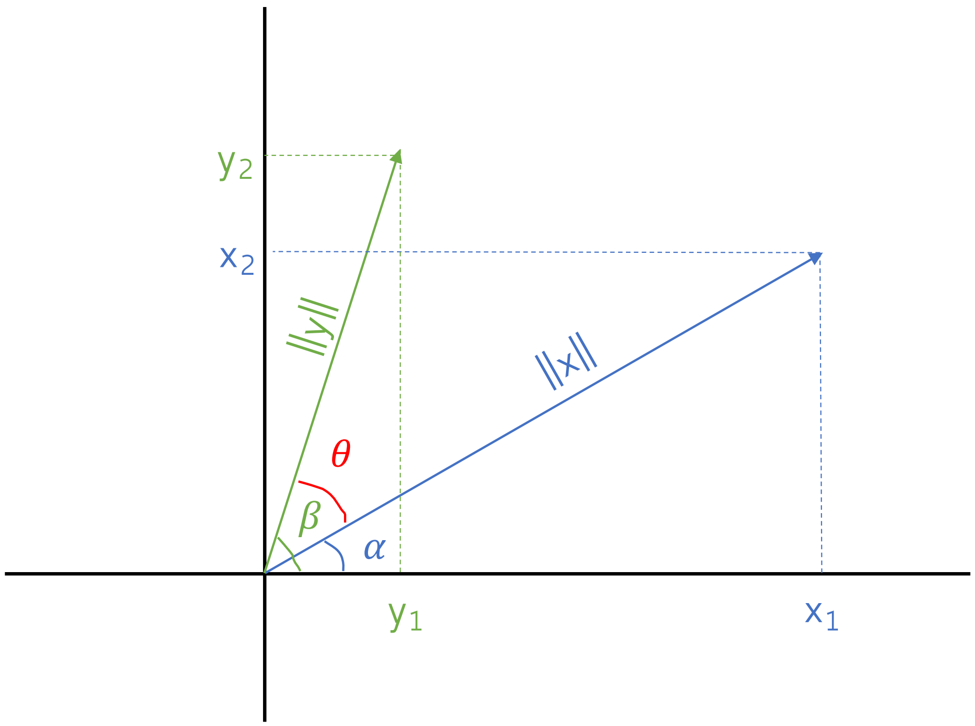


$$\theta = \beta - \alpha$$

$$x = (\underbrace{\|x\|\cos\alpha}_{x_1}, \underbrace{\|x\|\sin\alpha}_{x_2})$$

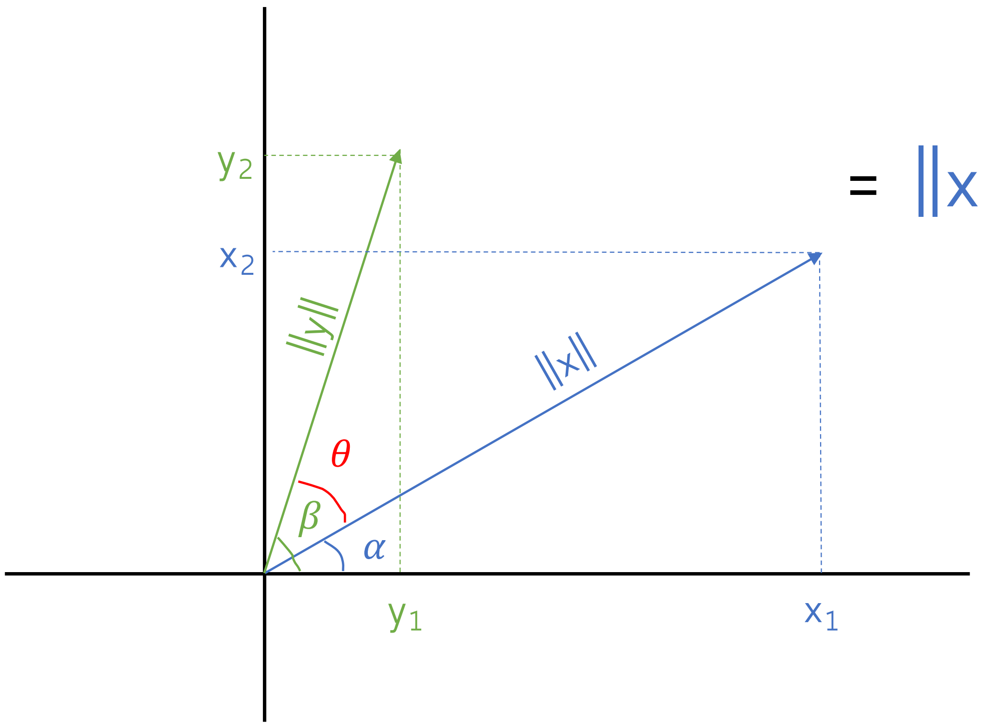
$$y = (\underbrace{\|y\|\cos\beta}_{y_1}, \underbrace{\|y\|\sin\beta}_{y_2})$$

Cosine Similarity: 2-dimensional Case



$$x \cdot y = x_1 y_1 + x_2 y_2 =$$

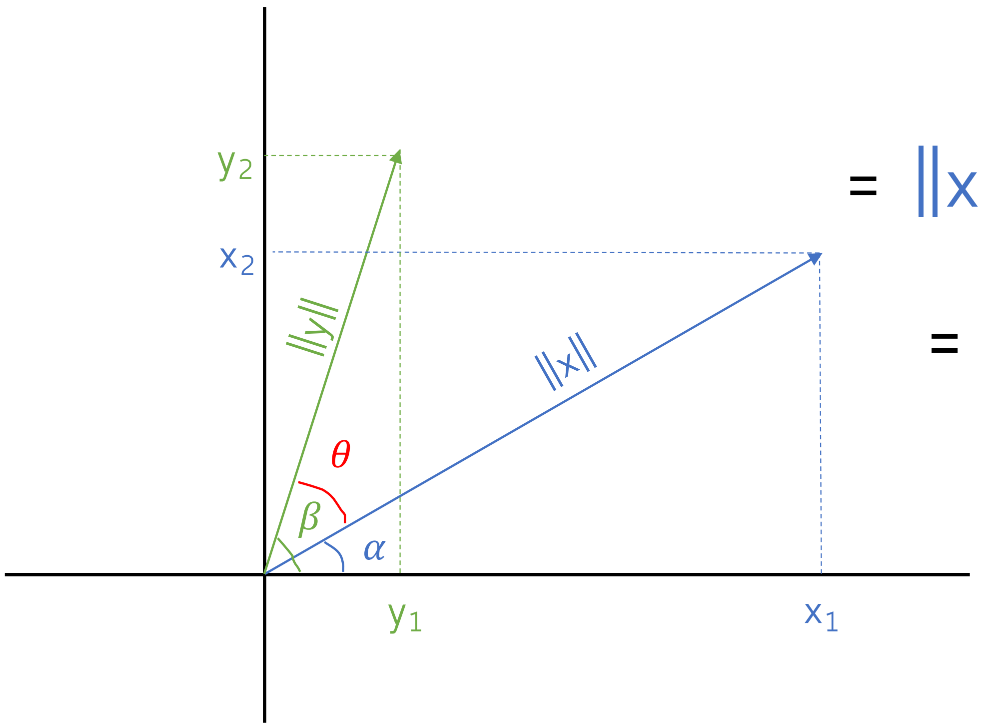
Cosine Similarity: 2-dimensional Case



$$x \cdot y = x_1 y_1 + x_2 y_2 =$$

$$= \|x\| \cos\alpha \|y\| \cos\beta + \|x\| \sin\alpha \|y\| \sin\beta$$

Cosine Similarity: 2-dimensional Case

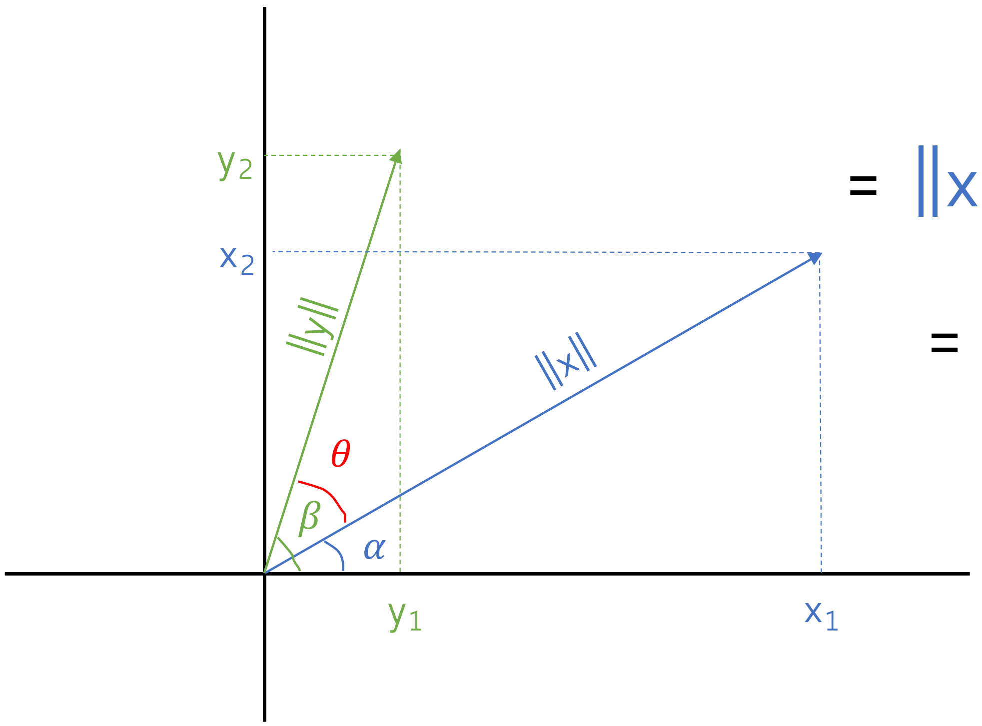


$$x \cdot y = x_1 y_1 + x_2 y_2 =$$

$$= \|x\|\cos\alpha\|y\|\cos\beta + \|x\|\sin\alpha\|y\|\sin\beta$$

$$= \|x\|\|y\|(\cos\alpha\cos\beta + \sin\alpha\sin\beta)$$

Cosine Similarity: 2-dimensional Case

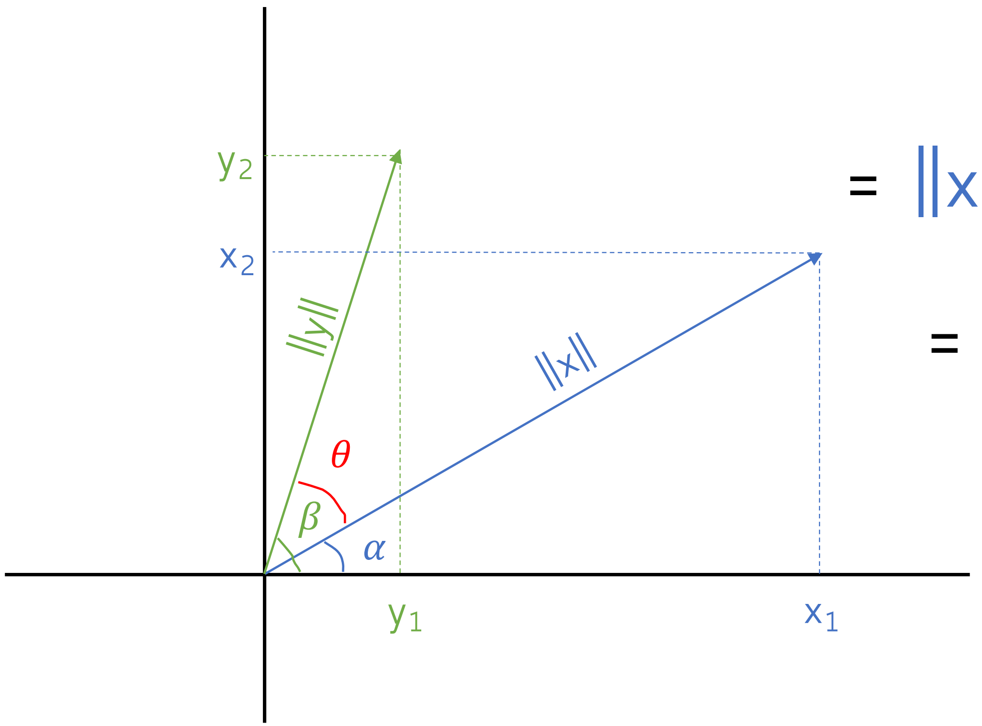


$$\mathbf{x} \cdot \mathbf{y} = x_1 y_1 + x_2 y_2 =$$

$$= \|x\|\cos\alpha\|y\|\cos\beta + \|x\|\sin\alpha\|y\|\sin\beta$$

$$= \|x\|\|y\|\underbrace{(\cos\alpha\cos\beta + \sin\alpha\sin\beta)}_{\cos(\underbrace{\beta - \alpha}_{\theta})}$$

Cosine Similarity: 2-dimensional Case



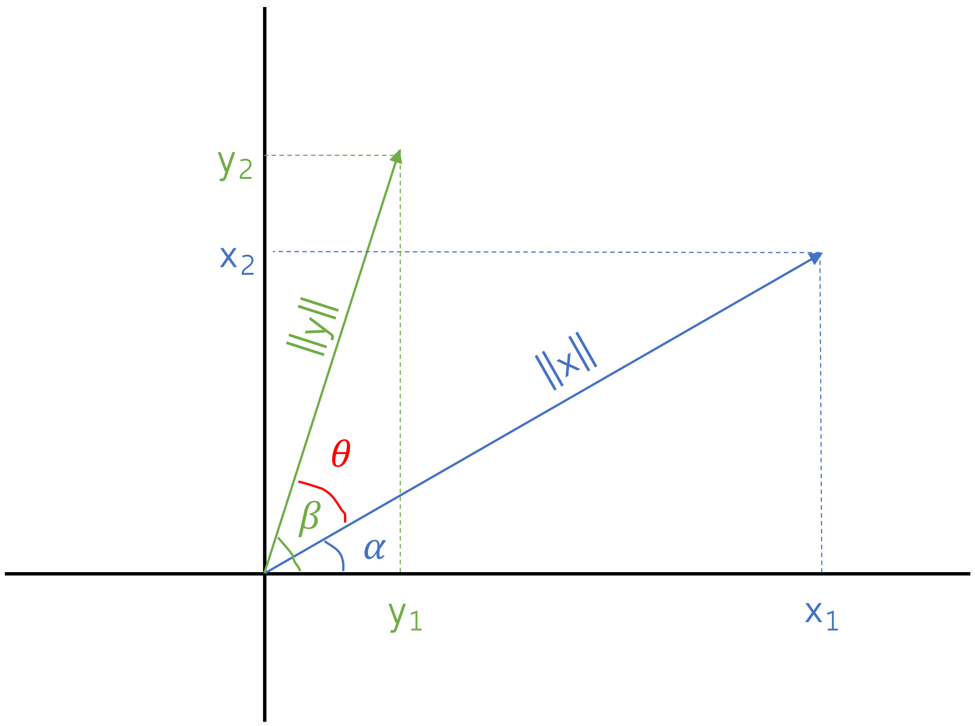
$$x \cdot y = x_1 y_1 + x_2 y_2 =$$

$$= \|x\|\cos\alpha\|y\|\cos\beta + \|x\|\sin\alpha\|y\|\sin\beta$$

$$= \|x\|\|y\|(\underbrace{\cos\alpha\cos\beta + \sin\alpha\sin\beta}_{\cos(\underbrace{\beta - \alpha}_{\theta})})$$

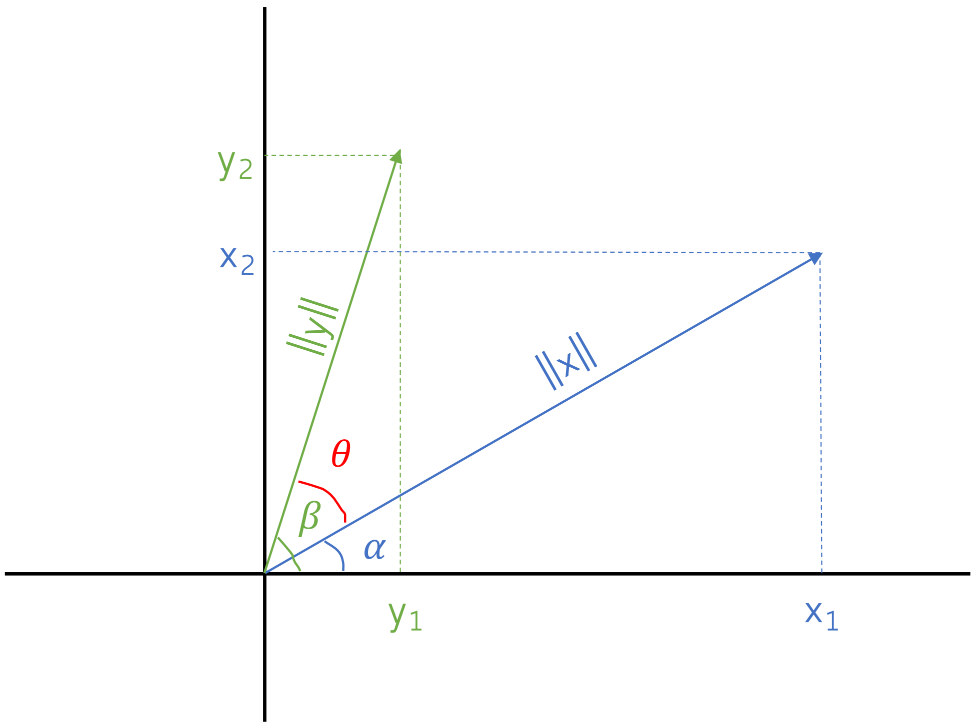
$$\boxed{x \cdot y = \|x\|\|y\|\cos\theta}$$

Cosine Similarity: 2-dimensional Case



$$x \cdot y = \|x\| \|y\| \cos\theta$$

Cosine Similarity: 2-dimensional Case



$$x \cdot y = \|x\|\|y\| \cos\theta$$

$$\boxed{\cos\theta = x \cdot y / \|x\|\|y\|}$$

Cosine Similarity: *d*-dimensional Case

- Computed as in the case of 2-dimensional vectors

Cosine Similarity: d-dimensional Case

- Computed as in the case of 2-dimensional vectors
- If two d-dimensional vectors are not collinear then they span a 2-dimensional plane $E \subset \mathbb{R}^d$

Cosine Similarity: d-dimensional Case

- Computed as in the case of 2-dimensional vectors
- If two d-dimensional vectors are not collinear then they span a 2-dimensional plane $E \subset \mathbb{R}^d$
- This plane E inherits the dot product in $\mathbb{R}^d$ and so becomes an ordinary Euclidean plane

Cosine Similarity: d-dimensional Case

- Computed as in the case of 2-dimensional vectors
- If two d-dimensional vectors are not collinear then they span a 2-dimensional plane $E \subset \mathbb{R}^d$
- This plane E inherits the dot product in $\mathbb{R}^d$ and so becomes an ordinary Euclidean plane
- The angles in this plane are related to the dot product as they are in 2-dimensional vector geometry

Jaccard Index (Coefficient)

Measures similarity between finite sample sets

Jaccard Index (Coefficient)

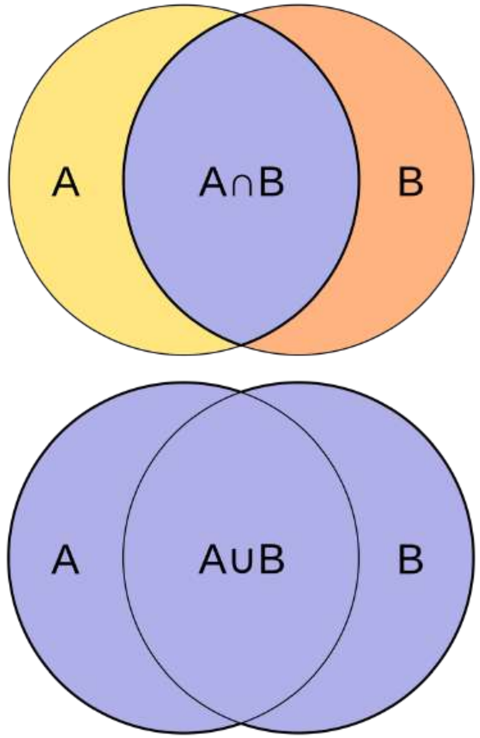
Measures similarity between finite sample sets

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

$$J(A,B) = 1 \text{ if } A = B = \emptyset$$

$$0 \le J(A,B) \le 1$$

Jaccard Index (Coefficient): Interpretation



source: Wikipedia

Jaccard Distance

Complementary to the Jaccard coefficient

$$\delta_J(A, B) = 1 - J(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

This distance is a metric on the collection of all finite sets

Take-Home Message of Today

- Vertical vs. Horizontal scale of data:
  - number of input data points vs. high-dimensional representation

Take-Home Message of Today

- Vertical vs. Horizontal scale of data:
  - number of input data points vs. high-dimensional representation
- Many big data tasks require computing "similarity" between domain items

Take-Home Message of Today

- Vertical vs. Horizontal scale of data:
  - number of input data points vs. high-dimensional representation
- Many big data tasks require computing "similarity" between domain items
- Different similarity measures:
  - Euclidean, Cosine, Jaccard, etc.

Take-Home Message of Today

- Vertical vs. Horizontal scale of data:
  - number of input data points vs. high-dimensional representation
- Many big data tasks require computing "similarity" between domain items
- Different similarity measures:
  - Euclidean, Cosine, Jaccard, etc.
- We'll see the notion of similarity can be flawed in high-dimensional spaces