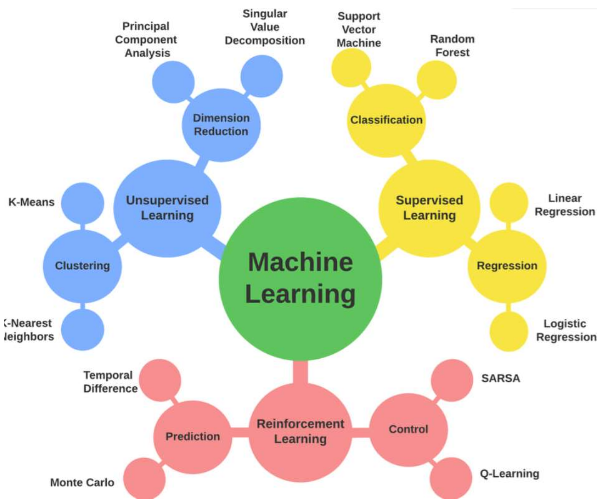
Các phương pháp học máy
Machine learning methods

4 TC: 2 LT – 2 TH

Giảng viên: **Tạ Hoàng Thắng**

tahoangthang@gmail.com

0975399307



Các phương pháp học máy
Machine learning methods

Bổ sung kiến thức:

- Đại số tuyến tính cơ bản

Đại số tuyến tính cơ bản

Scalars

Most everyday mathematics **consists of manipulating numbers** one at a time. Formally, we call these values scalars.

- For example, the temperature in Palo Alto is a balmy 72 degrees Fahrenheit.
  - What else?

Denote scalars by **ordinary lower-cased letters** (e.g., $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$) and the space of all (continuous) real-valued scalars by **R**.

- The expression $x \in \mathrm{R}$ is a formal way to say that x is a real-valued scalar.

Đại số tuyến tính cơ bản

Scalars

```
import torch
x = torch.tensor(3.0)
y = torch.tensor(2.0)
# x + y, x * y, x / y, x**y
```

```
(tensor(5.), tensor(6.), tensor(1.5000), tensor(9.))
```

Đại số tuyến tính cơ bản

Vectors

A vector can be thought as a fixed-length array of scalars.

- Denote vectors by bold lowercase letters, (e.g., **x**, **y**, and **z**)

```
x = torch.arange(3)
# tensor([0, 1, 2])
```

Đại số tuyến tính cơ bản

Vectors

A vector can be thought as a fixed-length array of scalars.

- Refer to an element of a vector by using a subscript.
  - For example, x_2 denotes the second element of **x**.

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

Đại số tuyến tính cơ bản

Vectors

```
x = torch.arange(3) # tensor([0, 1, 2])

x[2] # tensor(2)

len(x) # 3

x.shape # torch.Size([3])
```

Đại số tuyến tính cơ bản

Vectors

```
x = torch.arange(3) # tensor([0, 1, 2])
x[2] # tensor(2)
len(x)  # 3
x.shape  # torch.Size([3])
```

Đại số tuyến tính cơ bản

Matrices

Scalars are 0 th-order tensors and vectors are 1 st-order tensors, matrices are 2 nd-order tensors.

- Denote matrices by **bold capital letters** (e.g., **X, Y, and Z**), and represent them in code by tensors with two axes.
- The expression **A** ∈ R $m{\times}n$ indicates that a matrix **A** contains $m \times n$ real-valued scalars, arranged as m rows and n columns.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}.$$

Đại số tuyến tính cơ bản

Matrices

```
A = torch.arange(6).reshape(3, 2)
tensor([[0, 1], [2, 3], [4, 5]])

#  tensor([  [0, 1],
             [2, 3],
             [4, 5]  ])
```

Đại số tuyến tính cơ bản

Matrices

Signify **a matrix A's transpose by A^T and if $B = A^T$**, then $b_{ij} = a_{ij}$ for all i and j.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}. \qquad \mathbf{A}^{\top} = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{bmatrix}.$$

Đại số tuyến tính cơ bản

Matrices

```
A = torch.arange(6).reshape(3, 2)
#  tensor([[0, 1], [2, 3], [4, 5]])

print(A.T)
#  tensor([[0, 2, 4], [1, 3, 5]])
```

Đại số tuyến tính cơ bản

Tensors

A **tensor** is a **multi-dimensional data structure used for storing and processing data**. Tensors can be thought of as generalizations of scalars, vectors, and matrices to higher dimensions.

1. **0-D Tensor (Scalar)**: A single number, e.g., `3` or `-2.5`.
2. **1-D Tensor (Vector)**: A one-dimensional array of numbers, e.g., `[1, 2, 3]`.
3. **2-D Tensor (Matrix)**: A two-dimensional array, e.g., a matrix `[[1, 2], [3, 4]]`.
4. **3-D Tensor and Higher**: Tensors with more dimensions. For instance, a color image can be represented as a 3-D tensor with dimensions for height, width, and the number of color channels (e.g., RGB). A 4-D tensor might represent a batch of images, with dimensions for batch size, height, width, and color channels.

Đại số tuyến tính cơ bản

Tensors

A **tensor** is a **multi-dimensional data structure used for storing and processing data**. Tensors can be thought of as generalizations of scalars, vectors, and matrices to higher dimensions.

1. **0-D Tensor (Scalar)**: A single number, e.g., `3` or `-2.5`.
2. **1-D Tensor (Vector)**: A one-dimensional array of numbers, e.g., `[1, 2, 3]`.
3. **2-D Tensor (Matrix)**: A two-dimensional array, e.g., a matrix `[[1, 2], [3, 4]]`.
4. **3-D Tensor and Higher**: Tensors with more dimensions. For instance, a color image can be represented as a 3-D tensor with dimensions for height, width, and the number of color channels (e.g., RGB). A 4-D tensor might represent a batch of images, with dimensions for batch size, height, width, and color channels.

Đại số tuyến tính cơ bản

Tensors

```
torch.arange(24).reshape(2, 3, 4)
```

```
tensor([[[ 0,  1,  2,  3],
         [ 4,  5,  6,  7],
         [ 8,  9, 10, 11]],

        [[12, 13, 14, 15],
         [16, 17, 18, 19],
         [20, 21, 22, 23]]])
```

Đại số tuyến tính cơ bản

Tensors

```
torch.arange(24).reshape(2, 3, 4)
```

```
tensor([[[ 0,  1,  2,  3],
         [ 4,  5,  6,  7],
         [ 8,  9, 10, 11]],

        [[12, 13, 14, 15],
         [16, 17, 18, 19],
         [20, 21, 22, 23]]])
```

Đại số tuyến tính cơ bản

Elementwise operations

- refer to mathematical operations that are performed on **corresponding elements** of two or more tensors or arrays.

```
A = torch.arange(6, dtype=torch.float32).reshape(2, 3)
B = A.clone()  # Assign a copy of A to B by allocating new memory
A, A + B
```

```
(tensor([[0., 1., 2.],
         [3., 4., 5.]]),
 tensor([[ 0.,  2.,  4.],
         [ 6.,  8., 10.]]))
```

Đại số tuyến tính cơ bản

Hadamard product

The **elementwise product of two matrices** is called their Hadamard product (denoted $\odot$).

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \dots & a_{1n}b_{1n} \\ a_{21}b_{21} & a_{22}b_{22} & \dots & a_{2n}b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{m1} & a_{m2}b_{m2} & \dots & a_{mn}b_{mn} \end{bmatrix}.$$

Đại số tuyến tính cơ bản

Reduction

- refers to the process of performing operations that **reduce the dimensionality** of tensors.

```
x = torch.arange(3, dtype=torch.float32)
x, x.sum()
```

```
(tensor([0., 1., 2.]), tensor(3.))
```

Đại số tuyến tính cơ bản

Reduction

```python
import torch

A = torch.arange(6, dtype=torch.float32).reshape(2, 3)
print(A, A.shape)
# tensor([[0., 1., 2.],
#          [3., 4., 5.]]) torch.Size([2, 3])

B = A.sum()
print(B, B.shape)
#tensor(15.)

B = A.sum(axis=0)
print(B, B.shape)
#tensor([3., 5., 7.]) torch.Size([3])

B = A.sum(axis=1)
print(B, B.shape)
#tensor([ 3., 12.]) torch.Size([2])
```

Đại số tuyến tính cơ bản

Dot product (Tích vô hướng)

- **dot product** of two vectors is a way of multiplying them together to produce a single number (a scalar).
  - **Calculated by taking the sum of the products of their corresponding components**.

For Two Vectors

Given two vectors:

- $\mathbf{a} = [a_1, a_2, \ldots, a_n]$
- $\mathbf{b} = [b_1, b_2, \ldots, b_n]$

Their dot product is calculated as:

$$\mathbf{a} \cdot \mathbf{b} = a_1 \cdot b_1 + a_2 \cdot b_2 + \cdots + a_n \cdot b_n$$

Đại số tuyến tính cơ bản

Dot product (Tích vô hướng)

```python
import torch

y = torch.ones(3, dtype = torch.float32)
print(y)
# tensor([1., 1., 1.])

x = torch.arange(3, dtype = torch.float32)
print(x)
# tensor([0., 1., 2.])

z = torch.dot(x, y)
print(z)
# tensor(3.)

z1 = torch.sum(x * y)
print(z1)
# tensor(3.)
```

Đại số tuyến tính cơ bản

Dot product (Tích vô hướng)

```python
import torch

y = torch.ones(3, dtype = torch.float32)
print(y)
# tensor([1., 1., 1.])

x = torch.arange(3, dtype = torch.float32)
print(x)
# tensor([0., 1., 2.])

z = torch.dot(x, y)
print(z)
# tensor(3.)

```

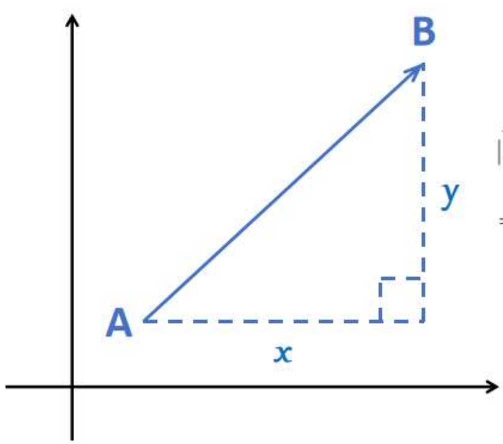
Đại số tuyến tính cơ bản

Euclidean Norm (L2 Norm)

For a vector **x**=[$x_1, x_2, \ldots, x_n$], the Euclidean norm is:

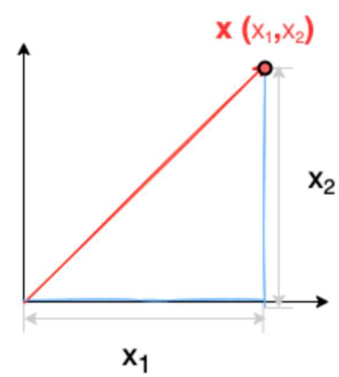
$$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

$$||x||_2 = \sqrt{x_1^2 + x_2^2}$$



$|\overrightarrow{AB}|^2 = x^2 + y^2$ (Định lý Pytago)

$\Rightarrow |\overrightarrow{AB}| = \sqrt{x^2 + y^2}$

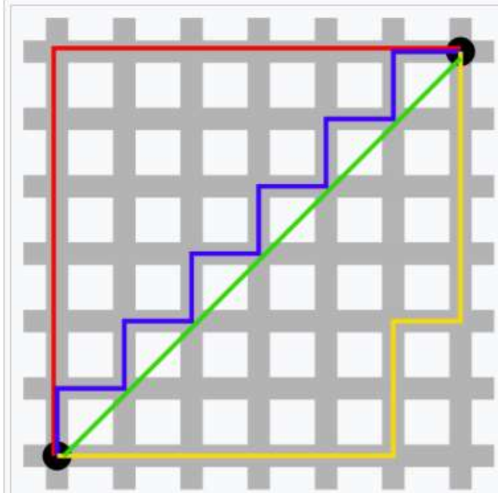


Đại số tuyến tính cơ bản

Manhattan Norm (L1 Norm)

For a vector **x**=[$x_1, x_2, \ldots, x_n$], the Manhattan norm is:

$$\|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_n|$$



So sánh giữa khoảng cách Euclid và khoảng cách Manhattan: Các đường màu đỏ, xanh lam, vàng biểu diễn khoảng cách Manhattan có cùng độ dài (12), trong khi đường màu xanh lục biểu diễn khoảng cách Euclid với độ dài 6×√2 ≈ 8.48.

Đại số tuyến tính cơ bản

Norms

```python
import torch

# Define a tensor
tensor = torch.tensor([3.0, 4.0])

# Compute the L1 norm (Manhattan norm)
l1_norm = torch.norm(tensor, p=1)
print(f"L1 norm (Manhattan norm): {l1_norm.item()}")
# L1 norm (Manhattan norm): 7.0

# Compute the L2 norm (Euclidean norm)
l2_norm = torch.norm(tensor, p=2)
print(f"L2 norm (Euclidean norm): {l2_norm.item()}")
# L2 norm (Euclidean norm): 5.0
```