Course Objectives:

* To learn the importance of linear algebra in practical problems and applications, in particular,
  - machine learning & pattern recognition
  - data mining & search engines
  - signal & image processing

* To learn important & useful concepts of linear algebra, e.g.,
  - linear transformations
  - bases & orthogonality
  - projections & least squares method
  - various matrix decompositions
    - LU, QR, eigenvalue, and SVD (singular value decomposition)

* To enhance your understanding of the above concepts & applications through the use of MATLAB
Motivation: Vector/Matrix Representation of Datasets

Example 1. Music Signals and Signal Compression ⇒ MATLAB Demo!

Example 2. Face Image Database ⇒ Figures

Example 3. Term-Document Matrices for Search

Example 4. Link Graphs of Webpages
Example 1: Music Signals & Signal Compression

Consider a very short signal consisting of 3 samples

\[
\mathbf{x} = [x_1, x_2, x_3]^T = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}
\]

This vector can be represented exactly as the following sum

\[
t = x_1 \cdot t_1 + x_2 \cdot t_2 + x_3 \cdot t_3
\]

In other words

\[
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + x_3 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}
\]

\[
\Leftrightarrow \mathbf{x} = x_1 \mathbf{e}_1 + x_2 \mathbf{e}_2 + x_3 \mathbf{e}_3
\]

Any \( \mathbf{x} \in \mathbb{R}^3 \) can be written exactly using the linear combination of \( \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \)
This can also be written as
\[
\begin{bmatrix}
e_1 \\
e_2 \\
e_3
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}
\]

However, depending on a signal, there may be a more efficient way to represent/approximate a given vector \( \mathbf{x} \).

e.g., consider the following vectors instead of \( e_1, e_2, e_3 \).

\[
\frac{1}{\sqrt{3}} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \Rightarrow t = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \equiv u_1
\]

\[
\frac{1}{\sqrt{12}} \begin{bmatrix} 1 \\ 0 \\ -\frac{1}{2} \end{bmatrix} \Rightarrow t = \frac{1}{\sqrt{12}} \begin{bmatrix} 1 \\ 0 \\ -\frac{1}{2} \end{bmatrix} \equiv u_2
\]

\[
\frac{1}{\sqrt{6}} \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \Rightarrow t = \frac{1}{\sqrt{6}} \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \equiv u_3
\]

Any \( \mathbf{x} \in \mathbb{R}^3 \) can be written this way!

\[
\mathbf{x} = a_1 u_1 + a_2 u_2 + a_3 u_3
\]

Given \( \mathbf{x} \), how to compute \( a_1, a_2, a_3 \)?

\[\Rightarrow \text{we'll learn how when we discuss "orthogonality"!}\]
We can also write this as

\[ \mathbf{x} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & 0 & -\frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \mathbf{U} \mathbf{a} \Rightarrow \mathbf{a} = \mathbf{U}^T \mathbf{x} \]

Note

But, we can only visualize such orthogonality up to 3D. The orthogonality can be generalized to any dimension \( \mathbb{R}^n \), \( n \geq 1 \)

For a real music signal with \( n \) huge, we can proceed similarly!
Input signal (say from your mp3 file).

Analog signal

Samples form a digital signal

Again, we can write such a digital signal $X = [x_1, x_2, \ldots, x_n]^T$ as a linear combination of basis vectors.

The basis I

The basis II

$$
\begin{align*}
& e_1 \quad \cdots \quad \uparrow \\
& e_2 \quad \cdots \quad \uparrow \\
& e_3 \quad \cdots \quad \uparrow \\
& \vdots \\
& e_n \quad \cdots \quad \uparrow \\
& X = x_1 e_1 + \cdots + x_n e_n \\
& \text{appropriately choosing the basis } \{u_1, \ldots, u_n\}
\end{align*}
$$

We can compress (or more precisely, compactly approximate) the input signal $X$!
For example, let’s use the so-called **Discrete Cosine Transform (DCT)** basis for \( \{u_1, \ldots, u_n\} \).

Then do the following operation:

1. Compute the coefficient vector 
   \[ a = [a_1, \ldots, a_n]^T. \]
2. For \( j = 1:n \),
   \[ \tilde{a}_j := \begin{cases} a_j & \text{if } |a_j| \geq \theta \text{ (a threshold)} \\ 0 & \text{otherwise} \end{cases} \]
3. Let \( \tilde{a} := [\tilde{a}_1, \ldots, \tilde{a}_n]^T \), and reconstruct an approximate signal
   \[ \tilde{x} = U \tilde{a} = \tilde{a}_1 u_1 + \cdots + \tilde{a}_n u_n \]

\[ n = 800.791 \]

\( \Rightarrow \) MATLAB Demo with my **music signal**
- \( \theta = 0.1 \Rightarrow 0.5\% \text{ of coeff's survived} \)
- \( \theta = 0.01 \Rightarrow 6\% \text{ of coeff's survived} \)

Also interesting to hear the approximation error \( x - \tilde{x} \)!

If you do the same experiment using \( \{e_1, \ldots, e_n\} \) instead of \( \{u_1, \ldots, u_n\} \), you’ll hear a huge difference!

That is, for a **usual music signal**, the DCT basis **is better than** the standard (or canonical) basis \( \{e_1, \ldots, e_n\} \).