# Lab 5: PCA and Autoencoders

August 2023

## 1 In This Lab

### 1.1 Topics to Cover

- PCA and Autoencoders

### 1.2 Requirements

- Python installed on your computer.
- Basic Python programming, Jupyter Notebook.
- Libraries: PyTorch, scikit-learn, numpy, matplotlib, seaborn

## 2 Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction technique used to transform high-dimensional data into a lower-dimensional space while preserving as much of the original data's variability as possible. PCA is based on finding the principal components, which are orthogonal vectors that capture the most significant directions of variation in the data.

### 2.1 Key Concepts

- **Variance Maximization:** PCA aims to maximize the variance of the projected data along the principal components. This ensures that the most important information is retained in the reduced dimensions.
- **Orthogonal Components:** Principal components are orthogonal to each other, meaning they are uncorrelated. The first principal component captures the most variance, the second captures the second most, and so on.
- **Eigenvalues and Eigenvectors:** PCA involves computing the eigenvalues and eigenvectors of the covariance matrix of the data. The eigenvectors represent the directions of the principal components, and the eigenvalues indicate the amount of variance captured along those directions.

- **Dimensionality Reduction:** After obtaining the eigenvalues and eigenvectors, you can project the original data onto a reduced-dimensional space defined by a subset of the principal components. This reduces the data's dimensionality while retaining important patterns.
- **Explained Variance:** The proportion of total variance captured by each principal component is called explained variance. It helps you understand how much information is retained in the reduced dimensions.

## 2.2 Steps of PCA

1. **Normalize Data:** Center the data by subtracting the mean of each feature from all data points.
2. **Compute Covariance Matrix:** Calculate the covariance matrix of the centered data.
3. **Eigenvalue Decomposition:** Compute the eigenvalues and eigenvectors of the covariance matrix.
4. **Sort and Select Components:** Sort the eigenvalues in descending order. The eigenvectors corresponding to the top $k$ eigenvalues are the principal components to retain.
5. **Projection:** Project the original data onto the lower-dimensional space defined by the selected principal components.

## 2.3 Applications

- Data Visualization: PCA is used to visualize high-dimensional data in 2D or 3D spaces for exploratory analysis.
- Noise Reduction: PCA can remove noise and redundancies by focusing on the principal components capturing the main variations.
- Feature Extraction: PCA is used as a preprocessing step to reduce feature dimensions before feeding data into machine learning models.

## 2.4 Dataset

- sample dataset (Iris)

## 2.5 Task 1:

1. Follow IrisPCA.ipynb for data visualization and steps

## 2.6 Report (use different dataset)

1. Follow the same steps as in IrisPCA with different dataset.

# 3 Autoencoder

An autoencoder is a neural network architecture used for unsupervised learning, dimensionality reduction, and feature learning. It consists of an encoder and a decoder network, which work together to learn a compressed representation of input data.

## 3.1 Key Components

- **Encoder:** The encoder network takes input data and maps it to a lower-dimensional representation, often referred to as the encoding or latent space. This reduces the dimensionality of the data and captures its essential features.
- **Decoder:** The decoder network takes the encoding and attempts to reconstruct the original input data. It mirrors the structure of the encoder but in reverse, aiming to generate output that closely matches the input.
- **Loss Function:** The autoencoder is trained to minimize the difference between the input and the reconstructed output. Common loss functions include mean squared error (MSE) or binary cross-entropy, depending on the nature of the data.
- **Bottleneck Layer:** The bottleneck layer, situated in the middle of the autoencoder, represents the compressed encoding. It is often smaller in dimension than the input and output layers, effectively reducing the data's dimensionality.

## 3.2 Applications

- **Dimensionality Reduction:** Autoencoders are used to reduce the dimensionality of high-dimensional data while retaining important features. The compressed representation can serve as input to other models or for data visualization.
- **Feature Learning:** Autoencoders can learn meaningful features from raw data without the need for manual feature engineering. The encoder learns to capture relevant patterns in the input.
- **Data Compression:** Autoencoders can compress data into a compact representation, making it suitable for storage and transmission in applications like image and audio compression.
- **Denoising:** By training on noisy data and using the original data as the target for reconstruction, autoencoders can be used for denoising, effectively removing noise from the input data.
- **Anomaly Detection:** Autoencoders can learn the normal patterns in data, and instances that have high reconstruction errors are often considered anomalies or outliers.

### 3.3 Dataset

- MNIST, CIFAR-10

### 3.4 Task 1:

1. Follow AutoencoderMNIST.ipynb for encoder and decoder

### 3.5 Report (use different dataset)

1. Use CIFAR-10 dataset and do the same steps as in previous example.