Principal Component Analysis

Nipun Batra

IIT Gandhinagar

October 30, 2025

The need for Dimensionality Reduction

- High-dimensional data is difficult to visualize and interpret.
- Many features may be correlated or redundant.
- Computational complexity increases with dimensions.

The need for Dimensionality Reduction

- High-dimensional data is difficult to visualize and interpret.
- Many features may be correlated or redundant.
- Computational complexity increases with dimensions.

Places where you will see dimensionality reduction

- Image compression and feature extraction in computer vision.
- Exploratory data analysis for visualizing high-dimensional datasets.
- Noise reduction and preprocessing for machine learning models.

Understanding Key Statistical Terms

Mean: The average value of a feature

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Understanding Key Statistical Terms

Mean: The average value of a feature

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Variance: Measures how spread out data is from the mean

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Understanding Key Statistical Terms

Mean: The average value of a feature

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Variance: Measures how spread out data is from the mean

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Higher variance = More spread out data = More information

Covariance: Measuring Relationships

Covariance: Measures how two variables change together

$$\text{Cov}(X, Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

Covariance: Measuring Relationships

Covariance: Measures how two variables change together

$$\mathrm{Cov}(X, Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

- Positive covariance: Variables increase together

Covariance: Measuring Relationships

Covariance: Measures how two variables change together

$$\text{Cov}(X, Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

- Positive covariance: Variables increase together
- Negative covariance: One increases, other decreases

Covariance Matrix: Contains all pairwise covariances

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \text{Cov}(X_1, X_2) & \cdots \\ \text{Cov}(X_2, X_1) & \sigma_2^2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

Covariance: Measuring Relationships

Covariance: Measures how two variables change together

$$\text{Cov}(X, Y) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

- Positive covariance: Variables increase together
- Negative covariance: One increases, other decreases
- Zero covariance: No linear relationship

Covariance Matrix: Contains all pairwise covariances

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \text{Cov}(X_1, X_2) & \cdots \\ \text{Cov}(X_2, X_1) & \sigma_2^2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

Eigenvectors and Eigenvalues: The Key to PCA

For a matrix A, an eigenvector v and eigenvalue λ satisfy:

$$Av = \lambda v$$

Eigenvectors and Eigenvalues: The Key to PCA

For a matrix A, an eigenvector v and eigenvalue λ satisfy:

$$Av = \lambda v$$

Intuition:

- Eigenvector: A special direction that doesn't change when matrix is applied

Eigenvectors and Eigenvalues: The Key to PCA

For a matrix A, an eigenvector v and eigenvalue λ satisfy:

$$Av = \lambda v$$

Intuition:

- Eigenvector: A special direction that doesn't change when matrix is applied
- Eigenvalue: How much the eigenvector is stretched or shrunk

Why we need them:

Eigenvectors and Eigenvalues: The Key to PCA

For a matrix A, an eigenvector v and eigenvalue λ satisfy:

$$Av = \lambda v$$

Intuition:

- Eigenvector: A special direction that doesn't change when matrix is applied
- Eigenvalue: How much the eigenvector is stretched or shrunk
- In PCA: Eigenvectors = Principal component directions

Why we need them:

Eigenvectors and Eigenvalues: The Key to PCA

For a matrix A, an eigenvector v and eigenvalue λ satisfy:

$$Av = \lambda v$$

Intuition:

- Eigenvector: A special direction that doesn't change when matrix is applied
- Eigenvalue: How much the eigenvector is stretched or shrunk
- In PCA: Eigenvectors = Principal component directions
- Eigenvalues = Amount of variance in those directions

Why we need them:

Eigenvectors and Eigenvalues: The Key to PCA

For a matrix A, an eigenvector v and eigenvalue λ satisfy:

$$Av = \lambda v$$

Intuition:

- Eigenvector: A special direction that doesn't change when matrix is applied
- Eigenvalue: How much the eigenvector is stretched or shrunk
- In PCA: Eigenvectors = Principal component directions
- Eigenvalues = Amount of variance in those directions

Why we need them:

- They reveal the natural axes of variation in the data

Eigenvectors and Eigenvalues: The Key to PCA

For a matrix A, an eigenvector v and eigenvalue λ satisfy:

$$Av = \lambda v$$

Intuition:

- Eigenvector: A special direction that doesn't change when matrix is applied
- Eigenvalue: How much the eigenvector is stretched or shrunk
- In PCA: Eigenvectors = Principal component directions
- Eigenvalues = Amount of variance in those directions

Why we need them:

- They reveal the natural axes of variation in the data
- Largest eigenvalues point to most important patterns

Housing Price Example: The Features

Suppose we're analyzing houses with many features:

Housing Price Example: The Features

Suppose we're analyzing houses with many features:
Size-related features: (highly correlated)

- Number of rooms
- Number of bathrooms
- Square footage
- Garage size

Housing Price Example: The Features

Suppose we're analyzing houses with many features:
Size-related features: (highly correlated)

- Number of rooms
- Number of bathrooms
- Square footage
- Garage size

Location-related features: (highly correlated)

- Number of schools nearby
- Crime rate
- Distance to city center
- Neighborhood income level

Housing Price Example: The Features

Suppose we're analyzing houses with many features:

Size-related features: (highly correlated)

- Number of rooms
- Number of bathrooms
- Square footage
- Garage size

Location-related features: (highly correlated)

- Number of schools nearby
- Crime rate
- Distance to city center
- Neighborhood income level

Problem: 8 features, but really just 2 underlying concepts!

Housing Example: Feature Correlations

Observation: Many features are redundant

Housing Example: Feature Correlations

Observation: Many features are redundant

- More rooms $\rightarrow$ More bathrooms $\rightarrow$ Larger square footage

Housing Example: Feature Correlations

Observation: Many features are redundant

- More rooms $\rightarrow$ More bathrooms $\rightarrow$ Larger square footage
- More schools $\rightarrow$ Lower crime $\rightarrow$ Higher neighborhood income

Covariance Matrix reveals:

Housing Example: Feature Correlations

Observation: Many features are redundant

- More rooms $\rightarrow$ More bathrooms $\rightarrow$ Larger square footage
- More schools $\rightarrow$ Lower crime $\rightarrow$ Higher neighborhood income

Covariance Matrix reveals:

- Size features have high positive covariances with each other

Housing Example: Feature Correlations

Observation: Many features are redundant

- More rooms $\rightarrow$ More bathrooms $\rightarrow$ Larger square footage
- More schools $\rightarrow$ Lower crime $\rightarrow$ Higher neighborhood income

Covariance Matrix reveals:

- Size features have high positive covariances with each other
- Location features have high covariances with each other

Housing Example: Feature Correlations

Observation: Many features are redundant

- More rooms $\rightarrow$ More bathrooms $\rightarrow$ Larger square footage
- More schools $\rightarrow$ Lower crime $\rightarrow$ Higher neighborhood income

Covariance Matrix reveals:

- Size features have high positive covariances with each other
- Location features have high covariances with each other
- But size and location features are nearly independent

Housing Example: PCA in Action

What PCA does:

Housing Example: PCA in Action

What PCA does:

- **PC1 (Principal Component 1):** Captures "overall size"
  - Combines rooms, bathrooms, square footage, garage
  - Explains, say, 60% of variance

Housing Example: PCA in Action

What PCA does:

- **PC1 (Principal Component 1):** Captures "overall size"
  - Combines rooms, bathrooms, square footage, garage
  - Explains, say, 60% of variance
- **PC2:** Captures "location quality"
  - Combines schools, crime rate, proximity, income
  - Explains, say, 30% of variance

Housing Example: PCA in Action

What PCA does:

- **PC1 (Principal Component 1):** Captures "overall size"
  - Combines rooms, bathrooms, square footage, garage
  - Explains, say, 60% of variance
- **PC2:** Captures "location quality"
  - Combines schools, crime rate, proximity, income
  - Explains, say, 30% of variance
- **Result:** 8 features reduced to 2 components

Housing Example: PCA in Action

What PCA does:

- **PC1 (Principal Component 1):** Captures "overall size"
  - Combines rooms, bathrooms, square footage, garage
  - Explains, say, 60% of variance
- **PC2:** Captures "location quality"
  - Combines schools, crime rate, proximity, income
  - Explains, say, 30% of variance
- **Result:** 8 features reduced to 2 components
- We've captured 90% of the information with 75% fewer features!

Housing Example: Interpretation

Original data:

$$\text{House}_i = [\text{rooms}, \text{bath}, \text{sqft}, \text{garage}, \text{schools}, \text{crime}, \text{distance}, \text{income}]$$

Housing Example: Interpretation

Original data:

$$\text{House}_i = [\text{rooms}, \text{bath}, \text{sqft}, \text{garage}, \text{schools}, \text{crime}, \text{distance}, \text{income}]$$

After PCA:

$$\text{House}_i = [\text{PC1}_{\text{size}}, \text{PC2}_{\text{location}}]$$

Housing Example: Interpretation

Original data:

$$\text{House}_i = [\text{rooms}, \text{bath}, \text{sqft}, \text{garage}, \text{schools}, \text{crime}, \text{distance}, \text{income}]$$

After PCA:

$$\text{House}_i = [\text{PC1}_{\text{size}}, \text{PC2}_{\text{location}}]$$

Benefits:

- Easier to visualize (2D scatter plot)

Housing Example: Interpretation

Original data:

$$\text{House}_i = [\text{rooms}, \text{bath}, \text{sqft}, \text{garage}, \text{schools}, \text{crime}, \text{distance}, \text{income}]$$

After PCA:

$$\text{House}_i = [\text{PC1}_{\text{size}}, \text{PC2}_{\text{location}}]$$

Benefits:

- Easier to visualize (2D scatter plot)
- Remove multicollinearity for regression models

Housing Example: Interpretation

Original data:

$$\text{House}_i = [\text{rooms}, \text{bath}, \text{sqft}, \text{garage}, \text{schools}, \text{crime}, \text{distance}, \text{income}]$$

After PCA:

$$\text{House}_i = [\text{PC1}_{\text{size}}, \text{PC2}_{\text{location}}]$$

Benefits:

- Easier to visualize (2D scatter plot)
- Remove multicollinearity for regression models
- Faster computation with fewer features

Housing Example: Interpretation

Original data:

$$\text{House}_i = [\text{rooms}, \text{bath}, \text{sqft}, \text{garage}, \text{schools}, \text{crime}, \text{distance}, \text{income}]$$

After PCA:

$$\text{House}_i = [\text{PC1}_{\text{size}}, \text{PC2}_{\text{location}}]$$

Benefits:

- Easier to visualize (2D scatter plot)
- Remove multicollinearity for regression models
- Faster computation with fewer features
- Core patterns are preserved

From Housing to Any Dataset

The PCA principle applies universally:

From Housing to Any Dataset

The PCA principle applies universally:

- **Images:** Thousands of pixels $\rightarrow$ Few Eigenfaces

From Housing to Any Dataset

The PCA principle applies universally:

- **Images:** Thousands of pixels $\rightarrow$ Few Eigenfaces
- **Genes:** Thousands of gene expressions $\rightarrow$ Key biological pathways

Common thread: Find the hidden, simpler structure in complex data

From Housing to Any Dataset

The PCA principle applies universally:

- **Images:** Thousands of pixels $\rightarrow$ Few Eigenfaces
- **Genes:** Thousands of gene expressions $\rightarrow$ Key biological pathways
- **Sensors:** Multiple correlated sensors $\rightarrow$ Underlying phenomena

Common thread: Find the hidden, simpler structure in complex data

From Housing to Any Dataset

The PCA principle applies universally:

- **Images:** Thousands of pixels $\rightarrow$ Few Eigenfaces
- **Genes:** Thousands of gene expressions $\rightarrow$ Key biological pathways
- **Sensors:** Multiple correlated sensors $\rightarrow$ Underlying phenomena
- **Text:** High-dimensional word vectors $\rightarrow$ Latent topics

Common thread: Find the hidden, simpler structure in complex data

What is PCA?

AIM: To find a lower-dimensional representation that captures maximum variance.

What is PCA?

AIM: To find a lower-dimensional representation that captures maximum variance.
KEY IDEA: Transform data to a new coordinate system where axes are ordered by variance.

What is PCA?

AIM: To find a lower-dimensional representation that captures maximum variance.

KEY IDEA: Transform data to a new coordinate system where axes are ordered by variance.

Examples:

Face Recognition: Reduce thousands of pixel features to a few principal components (Eigenfaces).

Gene Expression Analysis: Identify patterns across thousands of genes using a few components.

PCA Intuition

- Dataset with n samples and d features: $X \in \mathbb{R}^{n \times d}$

PCA Intuition

- Dataset with n samples and d features: $X \in \mathbb{R}^{n \times d}$
- Goal: Find k directions ($k < d$) that capture most variance

PCA Intuition

- Dataset with n samples and d features: $X \in \mathbb{R}^{n \times d}$
- Goal: Find k directions ($k < d$) that capture most variance
- These directions are the **principal components**

PCA Intuition

- Dataset with n samples and d features: $X \in \mathbb{R}^{n \times d}$
- Goal: Find k directions ($k < d$) that capture most variance
- These directions are the **principal components**
- First principal component: direction of maximum variance

PCA Intuition

- Dataset with n samples and d features: $X \in \mathbb{R}^{n \times d}$
- Goal: Find k directions ($k < d$) that capture most variance
- These directions are the **principal components**
- First principal component: direction of maximum variance
- Second principal component: direction of maximum remaining variance, orthogonal to the first

PCA Intuition

- Dataset with n samples and d features: $X \in \mathbb{R}^{n \times d}$
- Goal: Find k directions ($k < d$) that capture most variance
- These directions are the **principal components**
- First principal component: direction of maximum variance
- Second principal component: direction of maximum remaining variance, orthogonal to the first
- And so on...

PCA: Mathematical Formulation

- Center the data: $\tilde{X} = X - \bar{X}$

PCA: Mathematical Formulation

- Center the data: $\tilde{X} = X - \bar{X}$
- Compute covariance matrix: $\Sigma = \frac{1}{n-1}\tilde{X}^T\tilde{X}$

PCA: Mathematical Formulation

- Center the data: $\tilde{X} = X - \bar{X}$
- Compute covariance matrix: $\Sigma = \frac{1}{n-1}\tilde{X}^T\tilde{X}$
- Find eigenvectors and eigenvalues of Σ:

$$\Sigma v_i = \lambda_i v_i$$

PCA: Mathematical Formulation

- Center the data: $\tilde{X} = X - \bar{X}$
- Compute covariance matrix: $\Sigma = \frac{1}{n-1}\tilde{X}^T\tilde{X}$
- Find eigenvectors and eigenvalues of Σ:

$$\Sigma v_i = \lambda_i v_i$$

- Principal components are eigenvectors with largest eigenvalues

PCA: Mathematical Formulation

- Center the data: $\tilde{X} = X - \bar{X}$
- Compute covariance matrix: $\Sigma = \frac{1}{n-1}\tilde{X}^T\tilde{X}$
- Find eigenvectors and eigenvalues of Σ:

$$\Sigma v_i = \lambda_i v_i$$

- Principal components are eigenvectors with largest eigenvalues
- Projection onto k components:

$$Z = \tilde{X} W_k$$

where W_k contains top k eigenvectors

Variance Explained

- Each eigenvalue λ_i represents variance along component i

Variance Explained

- Each eigenvalue λ_i represents variance along component i
- Total variance: $\sum_{i=1}^{d} \lambda_i$

Variance Explained

- Each eigenvalue λ_i represents variance along component i
- Total variance: $\sum_{i=1}^{d} \lambda_i$
- Variance explained by component i:

$$\frac{\lambda_i}{\sum_{j=1}^{d} \lambda_j}$$

Variance Explained

- Each eigenvalue λ_i represents variance along component i
- Total variance: $\sum_{i=1}^{d} \lambda_i$
- Variance explained by component i:

$$\frac{\lambda_i}{\sum_{j=1}^{d} \lambda_j}$$

- Cumulative variance explained by first k components:

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{d} \lambda_j}$$

Variance Explained

- Each eigenvalue λ_i represents variance along component i
- Total variance: $\sum_{i=1}^{d} \lambda_i$
- Variance explained by component i:

$$\frac{\lambda_i}{\sum_{j=1}^{d} \lambda_j}$$

- Cumulative variance explained by first k components:

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{d} \lambda_j}$$

- Typically choose k such that cumulative variance $\geq 90\%$ or 95%

PCA Algorithm

Input: Data matrix $X \in \mathbb{R}^{n \times d}$, number of components k

PCA Algorithm

Input: Data matrix $X \in \mathbb{R}^{n \times d}$, number of components k

Steps:

1. Center the data: subtract mean from each feature

PCA Algorithm

Input: Data matrix $X \in \mathbb{R}^{n \times d}$, number of components k
Steps:

1. Center the data: subtract mean from each feature
2. Compute covariance matrix $\Sigma = \frac{1}{n-1}\tilde{X}^T\tilde{X}$

Output: Reduced data $Z \in \mathbb{R}^{n \times k}$

PCA Algorithm

Input: Data matrix $X \in \mathbb{R}^{n \times d}$, number of components k

Steps:

1. Center the data: subtract mean from each feature
2. Compute covariance matrix $\Sigma = \frac{1}{n-1}\tilde{X}^T\tilde{X}$
3. Compute eigendecomposition of Σ

Output: Reduced data $Z \in \mathbb{R}^{n \times k}$

PCA Algorithm

Input: Data matrix $X \in \mathbb{R}^{n \times d}$, number of components k
Steps:

1. Center the data: subtract mean from each feature
2. Compute covariance matrix $\Sigma = \frac{1}{n-1}\tilde{X}^T\tilde{X}$
3. Compute eigendecomposition of Σ
4. Sort eigenvectors by eigenvalues (descending)

Output: Reduced data $Z \in \mathbb{R}^{n \times k}$

PCA Algorithm

Input: Data matrix $X \in \mathbb{R}^{n \times d}$, number of components k
Steps:

1. Center the data: subtract mean from each feature
2. Compute covariance matrix $\Sigma = \frac{1}{n-1}\tilde{X}^T\tilde{X}$
3. Compute eigendecomposition of Σ
4. Sort eigenvectors by eigenvalues (descending)
5. Select top k eigenvectors as W_k

Output: Reduced data $Z \in \mathbb{R}^{n \times k}$

PCA Algorithm

Input: Data matrix $X \in \mathbb{R}^{n \times d}$, number of components k
Steps:

1. Center the data: subtract mean from each feature
2. Compute covariance matrix $\Sigma = \frac{1}{n-1}\tilde{X}^T\tilde{X}$
3. Compute eigendecomposition of Σ
4. Sort eigenvectors by eigenvalues (descending)
5. Select top k eigenvectors as W_k
6. Project data: $Z = \tilde{X}W_k$

Output: Reduced data $Z \in \mathbb{R}^{n \times k}$

PCA Properties

- **Linear transformation:** PCA finds linear combinations of original features

PCA Properties

- **Linear transformation:** PCA finds linear combinations of original features
- **Orthogonal components:** Principal components are mutually orthogonal

PCA Properties

- **Linear transformation:** PCA finds linear combinations of original features
- **Orthogonal components:** Principal components are mutually orthogonal
- **Maximizes variance:** Sequentially finds directions of maximum variance

PCA Properties

- **Linear transformation:** PCA finds linear combinations of original features
- **Orthogonal components:** Principal components are mutually orthogonal
- **Maximizes variance:** Sequentially finds directions of maximum variance
- **Minimizes reconstruction error:** Best k-dimensional linear approximation

PCA Properties

- **Linear transformation:** PCA finds linear combinations of original features
- **Orthogonal components:** Principal components are mutually orthogonal
- **Maximizes variance:** Sequentially finds directions of maximum variance
- **Minimizes reconstruction error:** Best k-dimensional linear approximation
- **Unsupervised:** Does not use label information

Choosing Number of Components

Several approaches:

- **Scree plot:** Plot eigenvalues and look for "elbow"

Choosing Number of Components

Several approaches:

- **Scree plot:** Plot eigenvalues and look for "elbow"
- **Cumulative variance:** Choose k such that $\geq 90\%$ or 95% variance explained

Choosing Number of Components

Several approaches:

- **Scree plot:** Plot eigenvalues and look for "elbow"
- **Cumulative variance:** Choose k such that $\geq 90\%$ or 95% variance explained
- **Kaiser criterion:** Keep components with $\lambda_i > 1$ (when data is standardized)

Choosing Number of Components

Several approaches:

- **Scree plot:** Plot eigenvalues and look for "elbow"
- **Cumulative variance:** Choose k such that $\geq$ 90% or 95% variance explained
- **Kaiser criterion:** Keep components with $\lambda_i > 1$ (when data is standardized)
- **Cross-validation:** If using PCA for downstream task, validate based on task performance

PCA Considerations

Advantages:

- Reduces dimensionality while preserving variance
- Removes multicollinearity
- Improves computational efficiency
- Aids visualization of high-dimensional data

PCA Considerations

Advantages:

- Reduces dimensionality while preserving variance
- Removes multicollinearity
- Improves computational efficiency
- Aids visualization of high-dimensional data

Limitations:

- Assumes linear relationships
- Sensitive to scaling (standardization recommended)
- Components may be hard to interpret
- Unsupervised: ignores class labels

PCA Applications

- **Image Processing:** Face recognition (Eigenfaces), image compression

PCA Applications

- **Image Processing:** Face recognition (Eigenfaces), image compression
- **Genomics:** Analyzing gene expression data, population structure

PCA Applications

- **Image Processing:** Face recognition (Eigenfaces), image compression
- **Genomics:** Analyzing gene expression data, population structure
- **Finance:** Risk modeling, portfolio analysis

PCA Applications

- **Image Processing:** Face recognition (Eigenfaces), image compression
- **Genomics:** Analyzing gene expression data, population structure
- **Finance:** Risk modeling, portfolio analysis
- **Signal Processing:** Noise reduction, feature extraction

PCA Applications

- **Image Processing:** Face recognition (Eigenfaces), image compression
- **Genomics:** Analyzing gene expression data, population structure
- **Finance:** Risk modeling, portfolio analysis
- **Signal Processing:** Noise reduction, feature extraction
- **Exploratory Data Analysis:** Visualizing high-dimensional datasets

PCA Applications

- **Image Processing:** Face recognition (Eigenfaces), image compression
- **Genomics:** Analyzing gene expression data, population structure
- **Finance:** Risk modeling, portfolio analysis
- **Signal Processing:** Noise reduction, feature extraction
- **Exploratory Data Analysis:** Visualizing high-dimensional datasets
- **Preprocessing:** Feature extraction before classification or regression

PCA vs Other Methods

- **PCA vs LDA:**
  - PCA: Unsupervised, maximizes variance
  - LDA: Supervised, maximizes class separability

PCA vs Other Methods

- **PCA vs LDA:**
  - PCA: Unsupervised, maximizes variance
  - LDA: Supervised, maximizes class separability
- **PCA vs t-SNE:**
  - PCA: Linear, preserves global structure
  - t-SNE: Nonlinear, preserves local structure, mainly for visualization

PCA vs Other Methods

- **PCA vs LDA:**
  - PCA: Unsupervised, maximizes variance
  - LDA: Supervised, maximizes class separability
- **PCA vs t-SNE:**
  - PCA: Linear, preserves global structure
  - t-SNE: Nonlinear, preserves local structure, mainly for visualization
- **PCA vs Autoencoders:**
  - PCA: Linear transformation
  - Autoencoders: Can learn nonlinear representations

Autoencoders

AIM: Learn nonlinear dimensionality reduction through neural networks.

Autoencoders

AIM: Learn nonlinear dimensionality reduction through neural networks.
ARCHITECTURE:

- **Encoder:** Maps input $x \in \mathbb{R}^d$ to latent representation $z \in \mathbb{R}^k$
  $z = f(x) = \sigma(Wx + b)$

Autoencoders

AIM: Learn nonlinear dimensionality reduction through neural networks.
ARCHITECTURE:

- **Encoder:** Maps input $x \in \mathbb{R}^d$ to latent representation $z \in \mathbb{R}^k$
  $z = f(x) = \sigma(Wx + b)$
- **Decoder:** Reconstructs input from latent representation
  $\hat{x} = g(z) = \sigma(W'z + b')$

Autoencoders

AIM: Learn nonlinear dimensionality reduction through neural networks.
ARCHITECTURE:

- **Encoder:** Maps input $x \in \mathbb{R}^d$ to latent representation $z \in \mathbb{R}^k$
  $z = f(x) = \sigma(Wx + b)$
- **Decoder:** Reconstructs input from latent representation
  $\hat{x} = g(z) = \sigma(W'z + b')$
- **Objective:** Minimize reconstruction error $\mathcal{L} = ||x - \hat{x}||^2$

Autoencoders: Key Properties

- **Nonlinear:** Can capture complex nonlinear relationships (unlike PCA)

Autoencoders: Key Properties

- **Nonlinear:** Can capture complex nonlinear relationships (unlike PCA)
- **Supervised training:** Trained with reconstruction objective

Autoencoders: Key Properties

- **Nonlinear:** Can capture complex nonlinear relationships (unlike PCA)
- **Supervised training:** Trained with reconstruction objective
- **Flexible architecture:** Can use deep networks for more expressive representations

Autoencoders: Key Properties

- **Nonlinear:** Can capture complex nonlinear relationships (unlike PCA)
- **Supervised training:** Trained with reconstruction objective
- **Flexible architecture:** Can use deep networks for more expressive representations
- **Bottleneck:** Latent dimension $k < d$ forces compression

Autoencoders: Key Properties

- **Nonlinear:** Can capture complex nonlinear relationships (unlike PCA)
- **Supervised training:** Trained with reconstruction objective
- **Flexible architecture:** Can use deep networks for more expressive representations
- **Bottleneck:** Latent dimension $k < d$ forces compression
- **Variants:** Denoising, variational, sparse, convolutional autoencoders

Autoencoders vs PCA

PCA:

- Linear transformation
- Closed-form solution
- Fast computation
- Global optimum guaranteed
- Interpretable components

Autoencoders:

- Nonlinear transformation
- Iterative optimization
- More computational cost
- Local minima possible
- Less interpretable

Autoencoders vs PCA

PCA:

- Linear transformation
- Closed-form solution
- Fast computation
- Global optimum guaranteed
- Interpretable components

Autoencoders:

- Nonlinear transformation
- Iterative optimization
- More computational cost
- Local minima possible
- Less interpretable

Note: Linear autoencoder with MSE loss learns PCA solution!

Autoencoder Applications

- **Image compression:** Learn compact representations of images

Autoencoder Applications

- **Image compression:** Learn compact representations of images
- **Anomaly detection:** High reconstruction error indicates anomalies

Autoencoder Applications

- **Image compression:** Learn compact representations of images
- **Anomaly detection:** High reconstruction error indicates anomalies
- **Denoising:** Remove noise by training on corrupted inputs

Autoencoder Applications

- **Image compression:** Learn compact representations of images
- **Anomaly detection:** High reconstruction error indicates anomalies
- **Denoising:** Remove noise by training on corrupted inputs
- **Feature learning:** Extract features for downstream tasks

Autoencoder Applications

- **Image compression:** Learn compact representations of images
- **Anomaly detection:** High reconstruction error indicates anomalies
- **Denoising:** Remove noise by training on corrupted inputs
- **Feature learning:** Extract features for downstream tasks
- **Generative modeling:** Variational autoencoders (VAEs) for generation

Autoencoder Applications

- **Image compression:** Learn compact representations of images
- **Anomaly detection:** High reconstruction error indicates anomalies
- **Denoising:** Remove noise by training on corrupted inputs
- **Feature learning:** Extract features for downstream tasks
- **Generative modeling:** Variational autoencoders (VAEs) for generation
- **Transfer learning:** Pre-train encoders on large unlabeled datasets

Summary

- PCA is a fundamental dimensionality reduction technique
- Finds orthogonal directions of maximum variance
- Based on eigendecomposition of covariance matrix
- Widely used for visualization, preprocessing, and compression
- Choose number of components based on variance explained or downstream task
- Consider data standardization before applying PCA
- Autoencoders extend to nonlinear dimensionality reduction