Unsupervised Learning

Dimensionality Reduction

Subhankar Mishra | smishra@niser.ac.in

The Modern Reality

High-dimensional data is everywhere:

- Modern ML algorithms (ensembles, neural networks) handle millions of features
- GPUs make high-dimensional computation feasible
- Dimensionality reduction used less than in the past

But we still need it for:

1. **Data visualization** - humans can only see 3D maximum
2. **Interpretable models** - when limited to simple algorithms
3. **Noise reduction** - removing redundant/correlated features

When Dimensionality Reduction Helps

Scenario 1: Data Visualization

- Need to understand high-dimensional data patterns
- Maximum 2D/3D plots for human interpretation
- Explore data structure and relationships

Scenario 2: Interpretable Models

- Limited to decision trees or linear regression , Need to understand which features matter, Simpler models with reduced dimensions

Scenario 3: Data Quality

- Remove redundant features, Reduce noise in data, Improve model interpretability

Four Main Techniques

1. Principal Component Analysis (PCA)

- Linear method, finds maximum variance directions
- Fast computation, interpretable results
- Standard choice for linear dimensionality reduction

2. t-Distributed Stochastic Neighbor Embedding (t-SNE)

- Non-linear method for visualization
- Preserves local neighborhood structure
- Computationally intensive, best for exploration

Four Main Techniques

3. Uniform Manifold Approximation and Projection (UMAP)

- Non-linear method, faster than t-SNE
- Balances local and global structure preservation
- Suitable for both visualization and preprocessing

4. Autoencoders

- Neural network approach
- Learns complex non-linear mappings
- Will be covered later

Principal Component Analysis (PCA)

Finding Directions of Maximum Variance

PCA: Core Intuition

Objective: Find new coordinate system based on data variance

Algorithm:

1. **First component:** Direction of highest variance in data
2. **Second component:** Orthogonal to first, second highest variance
3. **Third component:** Orthogonal to first two, third highest variance
4. **Continue** for all dimensions

Output: New axes (principal components) ranked by variance captured

PCA: Visual Example

Original 2D data → 1D projection

Step 1: Identify principal components

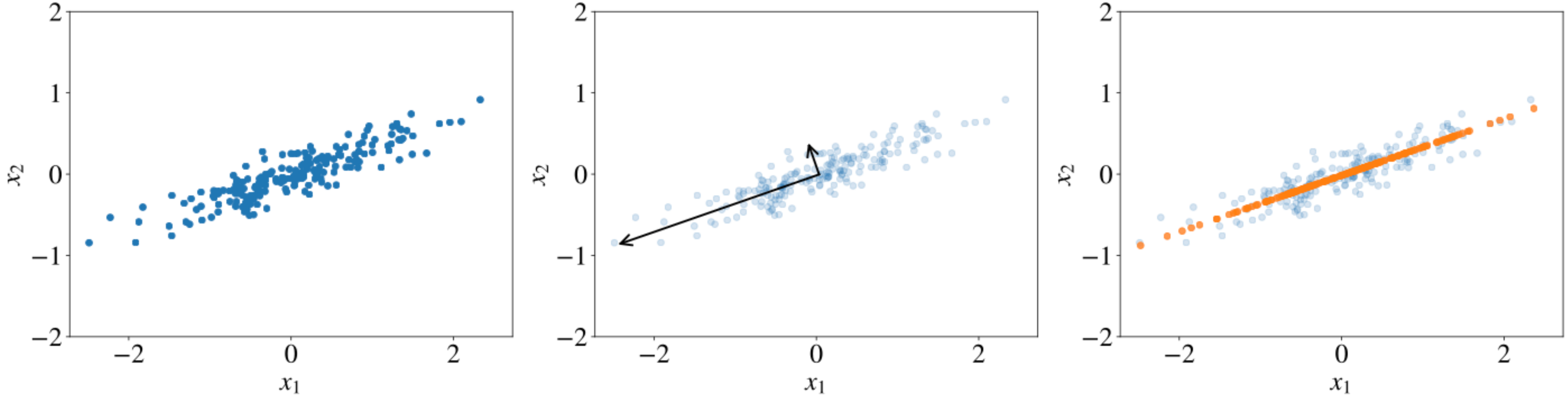
- PC1: Direction of maximum variance
- PC2: Orthogonal direction

Step 2: Project data onto first component

- Each point becomes single coordinate
- Dimensionality reduced from 2D → 1D

Key insight: Arrow length = variance in that direction

Subhankar Mishra | smishra@niser.ac.in



2
1
0
−1
−2
x_2
−2
0
2
x_1

PCA: Practical Benefits

Dimensionality reduction:

- Keep first $k < d$ principal components
- Discard components with low variance

Typical pattern:

- First 2-3 components capture 70-90% of variance
- Remaining components contain mostly noise

Visualization:

- Project high-dimensional data to 2D/3D
- Retain most important patterns in data

PCA: Mathematical Foundation

Objective: Find directions of maximum variance

Principal components are eigenvectors of covariance matrix

Variance captured by each component:

- PC1 captures most variance
- PC2 captures second most (orthogonal to PC1)
- Total variance = sum of all eigenvalues

Projection formula:

$$\mathbf{y} = \mathbf{W}^T(\mathbf{x} - \boldsymbol{\mu})$$

Where $\mathbf{W}$ contains first k principal components

t-SNE: The Visualization Specialist

Preserving Local Neighborhoods

t-SNE: Core Philosophy

Approach: Convert similarities to probabilities, then match distributions

Two-step process:

1. **Original space:** Define probability that points are neighbors
2. **Reduced space:** Match these probability distributions

Goal: Points close in high dimensions remain close in low dimensions

Primary use: Visualization and cluster structure exploration

t-SNE: Probability Definitions

Original space similarity (Gaussian):

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2/2\sigma_i^2)}$$

Symmetric version:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

Reduced space similarity (t-distribution):

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}$$

Why t-distribution? Heavy tails solve "crowding problem"

t-SNE: Optimization Process

Objective: Minimize KL divergence between P and Q

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Gradient descent update:

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}$$

Intuition:

- **Spring analogy:** Attractive and repulsive forces
- Points want to match their probability relationships

t-SNE: Key Properties

Advantages:

- **Visualization quality:** Clear cluster separation
- **Non-linear mapping:** Captures complex manifold structure
- **Local preservation:** Maintains neighborhood structure

Limitations:

- **Computational cost:** $O(n^2)$ complexity, slow for large datasets
- **Non-deterministic:** Different runs produce different results
- **Parameter sensitivity:** Perplexity choice affects results
- **Global structure loss:** Only local structure preserved

Appropriate use: Exploratory data visualization, not preprocessing

UMAP: Balanced Non-linear Approach

Preserving Local and Global Structure

UMAP: Core Philosophy

Approach: Preserve local neighborhoods in reduced space

Motivation:

- PCA captures only linear relationships
- Real data often has non-linear structure
- Local similarity important, but global structure also matters

UMAP method:

1. Define similarity metric in original space
2. Define same metric in reduced space
3. Minimize difference between similarity structures

UMAP: Similarity Metric

Combined similarity measure:

$$w(x_i, x_j) = w_i(x_i, x_j) + w_j(x_j, x_i) - w_i(x_i, x_j)w_j(x_j, x_i)$$

Individual similarity:

$$w_i(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j) - \rho_i}{\sigma_i}\right)$$

Where:

- $d(x_i, x_j)$ = Euclidean distance
- ρ_i = distance to closest neighbor
- σ_i = distance to k-th closest neighbor

UMAP: Optimization Process

Goal: Match similarity structures

Original space similarity: $w(x_i, x_j)$
Reduced space similarity: $w'(x'_i, x'_j)$

Cross-entropy loss:

$$C(w, w') = \sum_{i=1}^{N} \sum_{j=1}^{N} w(x_i, x_j) \ln \frac{w(x_i, x_j)}{w'(x'_i, x'_j)} + (1 - w(x_i, x_j)) \ln \frac{1 - w(x_i, x_j)}{1 - w'(x'_i, x'_j)}$$

Optimization: Use gradient descent to minimize $C(w, w')$

UMAP: Key Properties

Advantages:

- **Non-linear mapping:** Captures complex data structure
- **Local preservation:** Maintains neighborhood relationships
- **Computational efficiency:** Faster than t-SNE
- **Reproducibility:** More consistent results across runs

Properties:

- Similarity metric bounded [0, 1]
- Symmetric similarity: $w(x_i, x_j) = w(x_j, x_i)$
- Treats similarities as probability distributions

Method Comparison

Performance and Use Cases

Method Comparison: MNIST Example

Dataset: 70,000 handwritten digits, 10 classes

Cluster separation quality:

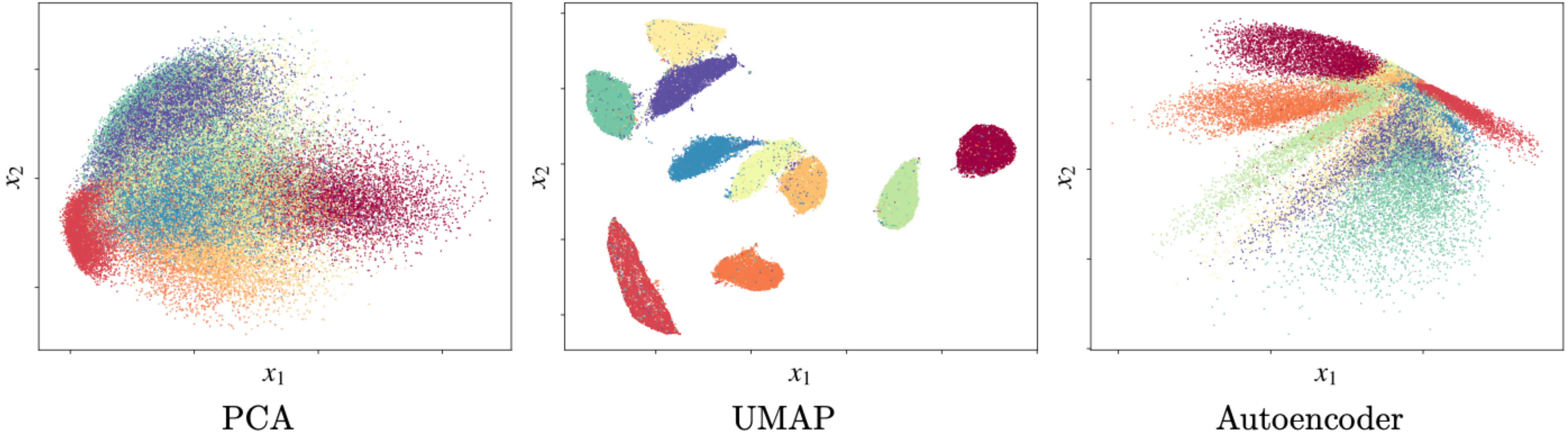
1. **t-SNE:** Clear cluster separation, computationally expensive
2. **UMAP:** Similar separation quality, faster computation
3. **PCA:** Linear projection, limited class separation

Computational performance:

- **PCA:** Fastest (seconds)
- **UMAP:** Medium speed (minutes)
- **t-SNE:** Slowest (hours for large datasets)

Note: Both t-SNE and UMAP achieve class separation without using labels

MNIST Example Comparison



Choosing the Right Method

Use PCA when:

- Need fast, simple solution
- Data has linear structure
- Want interpretable components
- Preparing data for other algorithms

Use t-SNE when:

- Primary goal is visualization
- Dataset is small-medium (<10k points)
- Want to explore cluster structure
- Don't need reproducible results

Choosing the Right Method

Use UMAP when:

- Need both speed and quality
- Large datasets (>10k points)
- Want to use reduced data for modeling
- Need reproducible results

Use Autoencoders when:

- Very complex non-linear relationships
- Need reconstruction capability
- Have GPU resources available

Practical Guidelines

Before dimensionality reduction:

1. **Scale features** - different units affect distance metrics
2. **Remove outliers** - can distort projections
3. **Consider feature selection** - remove irrelevant features first

Parameter tuning:

- **PCA:** Choose number of components (elbow method)
- **t-SNE:** Tune perplexity (5-50), learning rate (10-1000)
- **UMAP:** Tune number of neighbors, minimum distance
- **All methods:** Validate on downstream task

Validation Strategies

For visualization:

- Do clusters make domain sense?
- Are known relationships preserved?
- Can you explain the structure?

For model building:

- Cross-validate downstream model
- Compare performance vs. original features
- Check if interpretability improved

Common metrics:

- Explained variance ratio (PCA)

Common Pitfalls

Pitfall 1: "More dimensions is always better"

- **Reality:** Noise dimensions hurt performance
- **Solution:** Use validation to find optimal dimensions

Pitfall 2: "Linear methods are obsolete"

- **Reality:** PCA often works well and is interpretable
- **Solution:** Try simple methods first

Pitfall 3: "Visualization = analysis"

- **Reality:** 2D projections can be misleading
- **Solution:** Validate findings with quantitative methods

Summary: Key Takeaways

When to use dimensionality reduction:

- Data visualization needs
- Interpretable model requirements
- Noise reduction goals

Method selection:

- **PCA:** Linear structure, speed, interpretability
- **t-SNE:** Visualization, small datasets, cluster exploration
- **UMAP:** Non-linear structure, speed, general purpose
- **Autoencoders:** Complex patterns, reconstruction needs

Success factors:

Next Steps: Practice and Exploration

Immediate actions:

1. **Try all three methods** on same dataset
2. **Compare visualizations** - what do you see?
3. **Validate with downstream tasks**

Advanced topics:

- **Factor Analysis:** Probabilistic PCA
- **Non-negative Matrix Factorization:** Parts-based representation
- **Isomap:** Geodesic distance preservation
- **LLE:** Locally Linear Embedding

Remember: Dimensionality reduction is a tool, not a goal!

Questions for Discussion

1. When might high dimensions actually help your model?
2. How do you validate that a 2D visualization represents the real data structure?
3. What are the trade-offs between speed and quality in dimensionality reduction?
4. How would you explain PCA results to a non-technical stakeholder?

The best dimensionality reduction reveals meaningful patterns in your data.