Nonlinear Dimensionality Reduction

Nonlinear coordinates, distances, and local geometry

Kangning Cui
City University of Hong Kong (Dongguan)

Semester A, 2026/27

Upcoming dates

- **27 November:** Each project group submits one slide deck on Canvas.
- **30 November:** Group project presentations begin.
- **14 December, 09:00–12:00:** Closed-book final examination.

The final examination has five written questions, worth 20 marks each.

Project submission

The final project submission is a **single ZIP archive** uploaded to Canvas.

- **Report:** include the final report as a PDF (Portable Document Format) file.
- **Code:** include source code and instructions to run it.
- **GitHub:** if code is hosted online, include its **public repository link** in the ZIP archive.

All source code will be checked. Make sure the complete implementation is accessible.

Lecture focus

In earlier lectures, **principal component analysis (PCA)** reduced dimension using a flat subspace. Last week, kernels introduced nonlinear feature representations.

This lecture extends those ideas:

- Use **kernel principal component analysis (kernel PCA)** to summarize nonlinear features.
- Recover coordinates when only distances are available.
- Preserve useful relationships among local neighbors.

We will introduce each new method through the problem it is designed to solve.

Curved structure

Imagine marking points on a flat sheet. Their locations need only two coordinates, even if we bend the sheet into three-dimensional space.

A single flat projection may overlap different parts of the bent sheet. PCA cannot always preserve their original arrangement.

We first examine a rolled-sheet example, then study ways to retain its useful geometry.

Nonlinear geometry

CS5487 ▪ Semester A, 2026/27

Linear PCA

PCA represents the data using one flat subspace. Let $\boldsymbol{x} \in \mathbb{R}^d$ be an input, $\boldsymbol{\mu}$ the training mean, and $\boldsymbol{V}_r \in \mathbb{R}^{d \times r}$ the first r orthonormal principal directions.

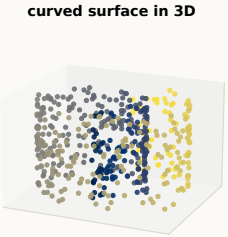
$$\boxed{\boldsymbol{z} = \boldsymbol{V}_r^{\mathsf{T}}(\boldsymbol{x} - \boldsymbol{\mu}), \qquad \boldsymbol{x} \approx \boldsymbol{\mu} + \boldsymbol{V}_r \boldsymbol{z}.}$$

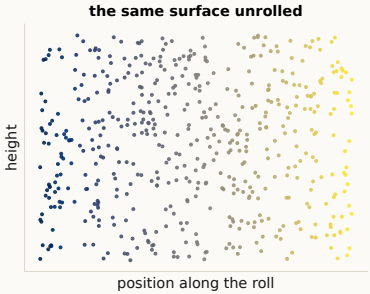
The reduced coordinates $\boldsymbol{z} \in \mathbb{R}^r$ work well for approximately flat data. Curved data may require a different representation.

Swiss roll

This example, called a **Swiss roll**, is a two-dimensional sheet rolled into three-dimensional space.

Colors follow position *along* the sheet. Points on adjacent turns may look close in 3D but be far apart on the surface.





Two distances

The Swiss roll introduces two ways to measure separation. Let $\boldsymbol{x}_i, \boldsymbol{x}_j$ be two points and let $d_{\text{surface}}(i, j)$ be their shortest distance *along the sheet*.

$$\underbrace{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2}_{\text{straight-line distance}} \quad \neq \quad \underbrace{d_{\text{surface}}(i, j)}_{\text{surface-path distance}} \ .$$

A curved, locally low-dimensional surface is an example of a **manifold**. Some methods preserve local relationships on this surface instead of direct 3D distances.

Preserved geometry

Before selecting an embedding method, decide which information the reduced coordinates should retain.

Information	Why it matters
Variance	Retain directions with substantial variation
Pairwise distances	Keep objects at comparable separations
Surface paths	Avoid shortcuts between curved layers
Local relationships	Keep nearby points in similar arrangements

Each method we introduce will preserve one of these properties.

Embedding size

An **embedding** assigns each input a shorter coordinate vector. For observation i, let $\boldsymbol{x}_i \in \mathbb{R}^d$ have d original features and $\boldsymbol{z}_i \in \mathbb{R}^r$ have $r < d$ reduced features.

$$\boxed{\boldsymbol{x}_i \in \mathbb{R}^d, \qquad \boldsymbol{z}_i \in \mathbb{R}^r.}$$

In the Swiss roll, points are recorded in 3D but lie on a sheet with two local coordinates.

Our goal is to retain useful relationships, not simply to make a two-dimensional plot.

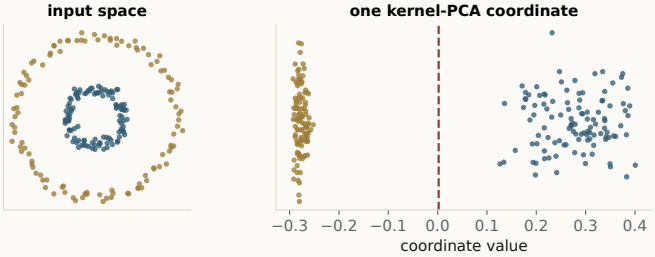
Kernel PCA

CS5487 ▪ Semester A, 2026/27

Concentric circles

A straight-line projection mixes points from two concentric rings.

Squared radius, $x_1^2 + x_2^2$, distinguishes the rings regardless of angle.



Radial example

Consider two groups of points $\boldsymbol{x} = (x_1, x_2)^\mathsf{T}$ on circles of radii 1 and 2. Their directions vary, but the squared radius $x_1^2 + x_2^2$ stays fixed within each group.

Group	Example points	Squared radius
Inner circle	$(1, 0),\ (0, -1)$	1
Outer circle	$(2, 0),\ (0, 2)$	4

Kernel PCA looks for high-variance directions after a nonlinear feature mapping, which can reveal such structure.

Feature-space PCA

Kernel PCA applies the familiar PCA objective after a nonlinear feature map. For n training observations $\boldsymbol{x}_i$, let $\phi(\boldsymbol{x}_i)$ be their mapped features.

Define the feature-space mean $\bar{\phi}$ and centered vectors $\widetilde{\phi}_i$:

$$\boxed{\bar{\phi} = \frac{1}{n}\sum_i \phi(\boldsymbol{x}_i), \qquad \widetilde{\phi}_i = \phi(\boldsymbol{x}_i) - \bar{\phi}.}$$

We want PCA on these centered features, but the map may be too large to construct explicitly.

Gram matrix

A kernel gives one inner product between two observations. The **Gram matrix** is simply the table of all these values.

For n inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, let

$$K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^\mathsf{T} \phi(\boldsymbol{x}_j).$$

For example, with $x_1 = 0$, $x_2 = 2$ and $k(x, z) = xz$:

$$\boxed{\boldsymbol{K} = \begin{bmatrix} k(0,0) & k(0,2) \\ k(2,0) & k(2,2) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix}.}$$

A row compares one observation with every training observation.

Why Gram matters

Ordinary PCA uses centered feature vectors. With nonlinear features, we may not know those coordinates, but a kernel gives their **inner products**.

The Gram matrix stores all pairwise values. After centering, it contains

$$\boxed{(K_c)_{ij} = \widetilde{\phi}_i^{\mathsf{T}} \widetilde{\phi}_j.}$$

Here $\widetilde{\phi}_i$ is the centered feature vector and $\boldsymbol{K}_c$ is its Gram matrix.
Eigendecomposing $\boldsymbol{K}_c$ lets us compute PCA coordinates without forming the nonlinear features.

Kernel centering

PCA measures variation around the training mean. The raw Gram matrix $\boldsymbol{K}$ uses uncentered features, so we must center its rows and columns.

For n training points, let $\boldsymbol{I}$ be the $n \times n$ identity and $\mathbf{1}$ the length-n vector of ones. Define

$$\boldsymbol{H} = \boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^{\mathsf{T}}, \qquad \boxed{\boldsymbol{K}_c = \boldsymbol{H}\boldsymbol{K}\boldsymbol{H}.}$$

The next example uses the same two observations as the Gram-matrix slide to show what changes.

Centering example

For training inputs $x_1 = 0$, $x_2 = 2$ and the linear kernel $k(x, z) = xz$, our Gram matrix was

$$\boldsymbol{K} = \begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix}, \qquad \boldsymbol{H} = \frac{1}{2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

Centering gives

$$\boxed{\boldsymbol{K}_c = \boldsymbol{H}\boldsymbol{K}\boldsymbol{H} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.}$$

This matches the inner products of centered input values $(-1, 1)$. The matrix now measures variation around the mean.

Kernel eigenproblem

Once the centered Gram matrix $\boldsymbol{K}_c$ is available, its eigenvectors provide the principal-component coordinates.

For component j, let $\kappa_j > 0$ be an eigenvalue and $\boldsymbol{u}_j \in \mathbb{R}^n$ a unit eigenvector:

$$\boxed{\boldsymbol{K}_c \boldsymbol{u}_j = \kappa_j \boldsymbol{u}_j, \qquad \|\boldsymbol{u}_j\|_2 = 1.}$$

We work with an $n \times n$ matrix rather than a potentially huge feature-space covariance matrix.

Eigenvalue scaling

The eigenvalue κ_j of the centered Gram matrix measures spread along component j.
If covariance uses the factor $1/n$ for n training observations, its corresponding feature-space variance is

$$\boxed{\lambda_j = \frac{\kappa_j}{n}.}$$

For example, $\kappa_j = 6$ with $n = 3$ gives $\lambda_j = 2$.

The eigenvector also tells us where individual observations lie along that component.

Training scores

An eigenvector gives relative positions of training observations; its eigenvalue determines their scale.

For component j, let z_{ij} denote the reduced coordinate of training observation i. Collect all n values in $\boldsymbol{z}_j$.

$$\boxed{\boldsymbol{z}_j = \sqrt{\kappa_j}\,\boldsymbol{u}_j.}$$

Here κ_j and $\boldsymbol{u}_j$ are the eigenpair of the centered Gram matrix.
A two-point example makes this formula concrete.

Two-point example

Use training values $x_1 = 0$ and $x_2 = 2$ with the linear kernel. Their centered Gram matrix is

$$\boldsymbol{K}_c = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

The positive eigenvalue is $\kappa = 2$. Choose the unit eigenvector $\boldsymbol{u} = (-1, 1)^\mathsf{T}/\sqrt{2}$.

$$\boxed{\boldsymbol{z} = \sqrt{2}\,\boldsymbol{u} = (-1, 1)^\mathsf{T}.}$$

These are exactly the centered scalar inputs. Kernel PCA reproduces ordinary PCA when the kernel is linear.

New observations

An embedding used for prediction must also transform new data.

Let $\boldsymbol{x}_*$ be a new observation and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ the *training* points. Calculate its kernel similarities with the training set:

$$\boxed{\boldsymbol{k}_* = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_*) & \cdots & k(\boldsymbol{x}_n, \boldsymbol{x}_*) \end{bmatrix}^\mathsf{T}.}$$

As with ordinary PCA, these comparisons must use the mean estimated from training data.

New-point centering

A new observation $\boldsymbol{x}_*$ must be centered against the *training* features, not against the test set.

Let $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Define: m_i as training Gram row i's mean; m_* as the mean similarity of the new point to training points; m as the training Gram's overall mean.

$$m_i = \tfrac{1}{n}\sum_j K_{ij}, \quad m_* = \tfrac{1}{n}\sum_j k(\boldsymbol{x}_j, \boldsymbol{x}_*), \quad m = \tfrac{1}{n^2}\sum_{j,\ell} K_{j\ell},$$

$$\widetilde{k}_{*i} = k(\boldsymbol{x}_i, \boldsymbol{x}_*) - m_i - m_* + m.$$

For component j, use $\boldsymbol{\alpha}_j = \boldsymbol{u}_j/\sqrt{\kappa_j}$ and $z_{*j} = \sum_i \alpha_{ij}\widetilde{k}_{*i}$.

New-point example

Continue with training values $(0, 2)$ and the linear kernel $k(x, z) = xz$. Their training mean is 1.

For the new value $x_* = 3$, the raw kernel vector is $\boldsymbol{k}_* = (0, 6)^\mathsf{T}$. Centering it with the *training* mean gives

$$\widetilde{\boldsymbol{k}}_* = (-2, 2)^\mathsf{T}.$$

For the component eigenvector from the previous example, the coefficients are $\boldsymbol{\alpha} = (-1/2, 1/2)^\mathsf{T}$. Thus

$$\boxed{z_* = \boldsymbol{\alpha}^\mathsf{T} \widetilde{\boldsymbol{k}}_* = 2.}$$

This agrees with ordinary PCA: $3 - 1 = 2$.

Pre-image problem

Ordinary PCA can reconstruct a point directly from its reduced coordinates. Kernel PCA first reconstructs a vector in its nonlinear *feature space*.

If $\hat{\phi}$ is that reconstructed feature vector, finding an original observation $\hat{\boldsymbol{x}}$ with

$$\boxed{\phi(\hat{\boldsymbol{x}}) \approx \hat{\phi}}$$

can be difficult. This is the **pre-image problem**.

It matters when reconstruction is required; a visualization or classifier may only need the reduced coordinates.

Kernel PCA settings

The kernel controls which nonlinear variations Kernel PCA can capture. Its settings must suit the task.

- **Kernel family:** linear, polynomial, or radial basis function (RBF).
- **Kernel scale:** for RBF, $\gamma > 0$ controls how quickly similarity decreases with distance.
- **Embedding dimension:** r is the number of components retained.

Use validation data to choose settings when the embedding is part of a predictive pipeline.

PCA comparison

Both methods seek high-variance coordinates, but they use different representations and different matrices.

	PCA	Kernel PCA
Geometry	Original features	Kernel feature space
Main matrix	Covariance	Centered Gram matrix
Directions	Feature vectors	Training-point weights
New data	Linear projection	Centered kernel values

Choose Kernel PCA only when its nonlinear geometry improves the intended representation or downstream task.

Multidimensional scaling

CS5487 ▪ Semester A, 2026/27

Distance input

Sometimes we know how far apart objects are, but not their original coordinates. We still want a low-dimensional map.

Classical multidimensional scaling (MDS) starts from distances d_{ij} between observations i, j and finds reduced coordinates $\boldsymbol{z}_i \in \mathbb{R}^r$ such that

$$\boxed{\|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2 \approx d_{ij}.}$$

To recover coordinates, MDS first converts distances into an inner-product table like the Gram matrix used in Kernel PCA.

Distance identity

To construct coordinates from distances, MDS needs a matrix of inner products. Squared Euclidean distances contain exactly this information.

For two points $\boldsymbol{x}_i, \boldsymbol{x}_j$, let $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$ and $\Delta_{ij} = d_{ij}^2$. Expanding gives

$$\boxed{\Delta_{ij} = \|\boldsymbol{x}_i\|_2^2 + \|\boldsymbol{x}_j\|_2^2 - 2\boldsymbol{x}_i^\mathsf{T}\boldsymbol{x}_j.}$$

After removing the mean contributions, the remaining information gives the centered Gram matrix.

Double centering

Kernel PCA begins with inner products. MDS begins with distances, but converts them to the same type of table.

For n observations, define the squared-distance matrix $\boldsymbol{\Delta} = [d_{ij}^2]$ and the centering matrix $\boldsymbol{H} = \boldsymbol{I} - \mathbf{1}\mathbf{1}^\mathsf{T}/n$.

$$\boxed{\boldsymbol{B} = -\frac{1}{2}\boldsymbol{H}\boldsymbol{\Delta}\boldsymbol{H}.}$$

The matrix $\boldsymbol{B}$ contains inner products of centered coordinates. Its leading eigenvectors provide the desired embedding.

MDS coordinates

From the recovered Gram matrix $\boldsymbol{B}$, we seek a small set of coordinates with approximately the same pairwise distances.

Let $\lambda_1, \ldots, \lambda_r$ be its largest *positive* eigenvalues, with orthonormal eigenvectors in $\boldsymbol{U}_r$.
Define $\boldsymbol{\Lambda}_r = \mathrm{diag}(\lambda_1, \ldots, \lambda_r)$.

$$\boxed{\boldsymbol{Z}_r = \boldsymbol{U}_r \boldsymbol{\Lambda}_r^{1/2}.}$$

Row i of $\boldsymbol{Z}_r$ is the reduced coordinate vector for observation i. An example follows.

Three-point distances

A short distance table shows how MDS works without original coordinates. For three observations, let $\mathbf{\Delta}$ contain their squared distances:

$$\mathbf{\Delta} = \begin{bmatrix} 0 & 1 & 9 \\ 1 & 0 & 4 \\ 9 & 4 & 0 \end{bmatrix}.$$

The distances between pairs $(1, 2)$, $(2, 3)$, and $(1, 3)$ are 1, 2, and 3.

These points can lie on a single line, so a one-dimensional embedding is sufficient.

Recovered Gram

For three points, suppose their squared-distance matrix is

$$\boldsymbol{\Delta} = \begin{bmatrix} 0 & 1 & 9 \\ 1 & 0 & 4 \\ 9 & 4 & 0 \end{bmatrix}.$$

After double centering, the recovered Gram matrix is

$$\boxed{\boldsymbol{B} = \frac{1}{9} \begin{bmatrix} 16 & 4 & -20 \\ 4 & 1 & -5 \\ -20 & -5 & 25 \end{bmatrix}.}$$

Its rank is one, matching the fact that these points can lie on a line.

Recovered positions

The preceding distances can be reproduced exactly in one dimension.

One set of centered coordinates is

$$\boxed{z_1 = -\tfrac{4}{3}, \qquad z_2 = -\tfrac{1}{3}, \qquad z_3 = \tfrac{5}{3}.}$$

Check their pairwise distances:

$$|z_1 - z_2| = 1, \quad |z_2 - z_3| = 2, \quad |z_1 - z_3| = 3.$$

MDS recovers positions up to translation, rotation, or reflection; these do not change distances.

Distance distortion

Three points at the corners of an equilateral triangle have pairwise distances $d_{12} = d_{23} = d_{13} = 2$.

Suppose a one-dimensional embedding places them at $z_1 = 0$, $z_2 = 1$, $z_3 = 2$.

Pair	Original distance	Embedded distance
$(1, 2)$	2	1
$(2, 3)$	2	1
$(1, 3)$	2	2

No three points on a line can keep all three distances equal to 2. Some distortion is unavoidable when reducing this triangle to one dimension.

MDS and PCA

PCA starts from observed feature coordinates. Classical MDS can start from their pairwise Euclidean distances instead.

Method	Starting information
PCA	Centered feature vectors
Classical MDS	Pairwise distances

For Euclidean data, their embeddings describe the same geometry up to rotation or reflection.

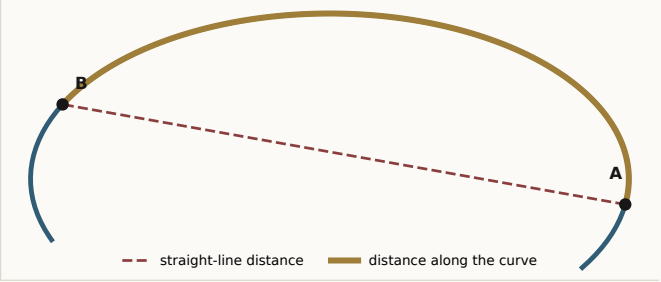
Isometric mapping

CS5487 ▪ Semester A, 2026/27

Surface paths

Isometric Mapping (ISOMAP) uses local graph paths to approximate distances along a curved surface.

The figure contrasts a direct shortcut with a route that follows the surface.



Local neighborhoods

ISOMAP cannot measure exact paths on an unknown curved surface. However, Euclidean distances are often reliable over short, local segments.

It connects nearby observations and assigns each edge its Euclidean length. Long distances are then estimated by joining these short segments.

A good neighborhood graph follows the surface without connecting distant folds directly.

Graph distances

Let each observation be a vertex of a neighborhood graph. An edge e joining nearby points has nonnegative length ℓ_e.

The **graph distance** $d_G(i,j)$ is the smallest sum of edge lengths among paths joining vertices i and j.

$$\boxed{d_G(i,j) = \min_{P \text{ joining } i,j} \sum_{e \in P} \ell_e.}$$

ISOMAP uses these graph distances to approximate travel along the curved surface.

Folded path

Consider four vertices A, B, C, D on the same chain, with edges AB, BC, and CD of length 1.

Edge	AB	BC	CD
Length	1	1	1

The only route from A to D uses all three edges, so

$$\boxed{d_G(A, D) = 1 + 1 + 1 = 3.}$$

Even if the straight-line distance is 0.3, the graph respects the chain's geometry.

Alternative paths

In a neighborhood graph, two points may have several connecting routes. ISOMAP uses the shortest one.

Suppose A and D are connected by two paths:

Route	Total edge length
A, B, D	$1 + 3 = 4$
A, C, D	$2 + 1 = 3$

Hence $\boxed{d_G(A, D) = 3}$. These shortest-path values become the distance input for MDS.

ISOMAP pipeline

Ordinary MDS uses a supplied distance table. ISOMAP first constructs a distance table that follows the manifold instead.

1. Connect each point to nearby observations.
2. Assign Euclidean lengths to local edges.
3. Compute the shortest graph path between every pair.
4. Apply classical MDS to those distances.

The quality of the neighborhood graph largely determines the result.

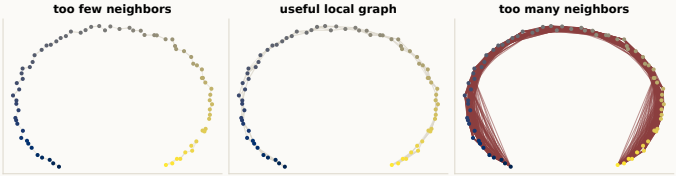
Neighborhood size

The figure shows why the number of neighbors matters.

Too few: disconnected graph.

Too many: shortcuts between folds.

Suitable: edges follow the surface.



Disconnected graphs

If two graph vertices have no connecting path, their shortest-path distance is infinite:

$$\boxed{d_G(i,j) = \infty \quad \text{when vertices } i, j \text{ are disconnected.}}$$

Classical MDS cannot directly use infinite entries in its distance matrix.

ISOMAP therefore needs a graph that connects the data we want to embed.

Shortcut edges

Too many neighbors may introduce a direct connection between different folds of a surface.

For the previous chain A, B, C, D, the graph path from A to D had length 3. Suppose we incorrectly add a direct edge of length 0.3.

$$\boxed{d_G(A, D) = 0.3 \quad \text{after the shortcut is added.}}$$

The graph now understates the true distance along the folded surface. The resulting embedding may be distorted.

ISOMAP comparison

MDS and ISOMAP both recover coordinates by eigendecomposition. They differ in the input distances.

Method	Distances used
Classical MDS	Direct pairwise distances
ISOMAP	Shortest paths on a neighbor graph

Use ISOMAP when straight-line distances cut across a curved surface.

ISOMAP limits

A graph only approximates the unknown manifold. Its quality depends on sampling and neighborhood choice.

- Sparse data may not provide reliable paths.
- Noise can create incorrect neighbor edges.
- Some surfaces cannot be flattened globally without distortion.

ISOMAP is useful when the graph follows the surface, not when the graph introduces artificial shortcuts.

Locally linear embedding

CS5487 ▪ Semester A, 2026/27

Local reconstruction

ISOMAP preserves distances along a graph. **Locally Linear Embedding (LLE)** instead preserves how each observation is reconstructed from nearby points.

Let $\boldsymbol{x}_i \in \mathbb{R}^d$ be observation i, let $\mathcal{N}(i)$ contain the indices of its neighbors, and let w_{ij} be the weight of neighbor j:

$$\boxed{\boldsymbol{x}_i \approx \sum_{j \in \mathcal{N}(i)} w_{ij} \boldsymbol{x}_j, \qquad \sum_{j \in \mathcal{N}(i)} w_{ij} = 1.}$$

These weights summarize local position without requiring global path distances.

LLE weights

LLE represents each point using nearby points rather than storing long paths between them.

Let $\boldsymbol{x}_i$ be the target observation, $\mathcal{N}(i)$ its neighbor indices, and w_{ij} the weight assigned to neighbor j.

$$\min_{w_{ij}} \left\| \boldsymbol{x}_i - \sum_{j \in \mathcal{N}(i)} w_{ij} \boldsymbol{x}_j \right\|_2^2, \quad \sum_j w_{ij} = 1.$$

These weights capture local position. Once learned, they remain fixed during the embedding step.

Midpoint example

Local reconstruction is simple on a line. Let the target be $x_i = 1$ and its two neighbors be $x_A = 0$, $x_B = 2$.

$$1 = w_A(0) + w_B(2), \qquad w_A + w_B = 1.$$

Solving gives

$$\boxed{w_A = w_B = \tfrac{1}{2}}, \qquad 1 = \tfrac{1}{2}(0) + \tfrac{1}{2}(2).$$

LLE tries to keep this midpoint relationship after embedding the points.

Weight preservation

The local reconstruction weights tell us where a point lies relative to its neighbors.

After fitting the weights w_{ij}, LLE assigns each point a reduced coordinate $\boldsymbol{z}_i \in \mathbb{R}^r$ and tries to keep

$$\boxed{\boldsymbol{z}_i \approx \sum_j w_{ij} \boldsymbol{z}_j.}$$

For example, a point halfway between two neighbors should remain approximately halfway between their embedded positions.

This differs from ISOMAP: LLE preserves local relationships rather than all graph distances.

LLE objective

How can one set of reduced coordinates preserve the weights for *all* observations?

With n embedded points $\boldsymbol{z}_i$ and fixed reconstruction weights w_{ij}, LLE minimizes

$$\sum_{i=1}^{n} \left\| \boldsymbol{z}_i - \sum_{j} w_{ij} \boldsymbol{z}_j \right\|_2^2 .$$

LLE also fixes the mean and scale of the embedding; otherwise all points could collapse to the same location.

The important idea is the local reconstruction relation, not the details of the numerical eigensolver.

Computing LLE

The reconstruction weights are now fixed. We need coordinates that keep these local relationships without placing every point at the same location.

Let $\boldsymbol{W} = [w_{ij}]$ contain the weights for all n observations, and let $\boldsymbol{I}$ be the $n \times n$ identity matrix.

$$\boxed{\boldsymbol{M} = (\boldsymbol{I} - \boldsymbol{W})^{\mathsf{T}}(\boldsymbol{I} - \boldsymbol{W}).}$$

LLE uses the r smallest nonconstant eigenvectors of $\boldsymbol{M}$ as coordinates in $\mathbb{R}^r$. This computes a layout while preserving the learned neighbor relations.

Unequal weights

The target need not be halfway between its neighbors. Let $x_i = 1$, with neighbors $x_A = 0$ and $x_B = 4$.

$$1 = w_A(0) + w_B(4), \qquad w_A + w_B = 1.$$

The solution is

$$\boxed{w_A = \tfrac{3}{4}, \qquad w_B = \tfrac{1}{4}.}$$

LLE seeks embedded coordinates satisfying $z_i \approx \frac{3}{4} z_A + \frac{1}{4} z_B$.

Local methods

ISOMAP and LLE both begin with neighbors, but they keep different information after reducing dimension.

Method	Structure preserved
ISOMAP	Shortest-path distances
LLE	Neighbor reconstruction weights

Both depend on the quality of the local neighborhood graph.

Embedding evaluation

CS5487 ▪ Semester A, 2026/27

Visual evidence

A two-dimensional embedding can look convincing even when important relationships have been distorted.

A visible gap between groups does *not* by itself establish that the groups represent meaningful structure.

Evaluate the representation according to the information the method was supposed to preserve, as well as the eventual task.

Neighbor quality

Manifold methods rely on local neighborhoods. We need to check whether these neighborhoods survive the embedding.

- **Trustworthiness:** points shown as close neighbors in the embedding should be close in the original data.
- **Continuity:** original neighbors should remain close after embedding.

These detect false neighbors and lost neighbors, respectively.

Neighbor example

Suppose B and C were the two nearest neighbors of observation A before reduction. The embedding instead places B and D closest to A.

	Original data	Embedding
Two neighbors of A	$\{B, C\}$	$\{B, D\}$

Here D is a **false neighbor**; C is a **lost neighbor**. Neighborhood metrics measure these kinds of changes.

Distance preservation

For methods that aim to keep distances, compare the original distance d_{ij} between observations i, j with the distance between their reduced coordinates $\boldsymbol{z}_i, \boldsymbol{z}_j$:

$$\boxed{d_{ij} \qquad \text{versus} \qquad \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2.}$$

For example, if $d_{12} = 2$ but the embedded distance is 1, that pair has distance error 1.

Local and global distance errors answer different questions.

Prediction

An embedding can preserve geometry while losing information needed for prediction.

Let x denote the original features and z the reduced features used by a classifier. The classifier's accuracy depends on both the embedding and the classifier.

If prediction is the goal, choose the embedding settings on validation data and evaluate the *complete pipeline* on unseen test observations.

Stability

A useful pattern should not disappear when we slightly change the data or a modeling choice.

Check whether nearby points remain neighbors after:

- small changes to the training sample or measurement noise;
- reasonable changes to kernel scale or graph neighborhood size.

Large changes signal that the embedding may be fragile.

Out-of-sample data

A visualization may only need training points. A deployed prediction system must also handle new observations.

- **PCA:** use the previously fitted mean and principal directions.
- **Kernel PCA:** compare a new point with training observations and use the fitted centering and components.
- **ISOMAP and LLE:** classical versions generally require an additional out-of-sample extension or refitting.

This practical requirement can influence method selection.

Computational scale

For n observations, a dense matrix of pairwise distances or kernel values contains n^2 numbers:

$$\boxed{\text{memory} = O(n^2).}$$

At $n = 10{,}000$, this means 10^8 entries, or about 800 MB using 8 bytes per value before overhead.

Eigendecomposition and graph shortest paths add further computational cost.

Landmarks

Exact methods often store many pairwise kernel or distance values. For large training sets, we can approximate them using representative examples.

The **Nyström approach** selects m landmark observations from n training points, with $m \ll n$, then uses the kernel similarities to those landmarks to approximate the full matrix.

This can reduce memory and computation, at the cost of approximation error.

Choosing a method

Choose an embedding method based on the geometric information your task needs.

Structure to preserve	Method
Linear variance	PCA
Nonlinear feature variance	Kernel PCA
Pairwise distances	Classical MDS
Surface-following paths	ISOMAP
Local reconstruction	LLE

If the representation serves prediction, compare the complete pipelines using validation data.

Course connections

CS5487 ▪ Semester A, 2026/27

Clustering

Different clustering methods define a group in different ways.

Method	Grouping principle
K-means	Low within-cluster squared distances
Gaussian mixture model (GMM)	Shared probability component
Mean shift	Shared density mode
Spectral clustering	Strong graph connections

Classification

Classification methods differ by the quantity they model or optimize. Here $\boldsymbol{x}$ denotes features and y the class label.

- **Gaussian classifiers and Naive Bayes:** use $p(\boldsymbol{x} \mid y)$ and class prior $p(y)$.
- **Logistic regression:** models $p(y \mid \boldsymbol{x})$ directly.
- **Support vector machine (SVM):** balances geometric margin and violation penalties.
- **Kernel SVM:** uses nonlinear feature geometry with the same margin principle.

Choose a method whose objective fits the task and available labels.

Representations

Across this week, dimensionality reduction has been a choice of which information to retain.

Method	Structure preserved
PCA	Linear variance and reconstruction
Kernel PCA	Feature-space variance
MDS	Pairwise distances
ISOMAP	Graph-path distances
LLE	Local reconstruction weights

Optimization

A method's objective suggests how to fit its parameters.

- Least squares can have a closed-form solution.
- K-means alternates point assignment and centroid updates.
- A Gaussian mixture model may use expectation-maximization (EM) for latent memberships.
- Logistic regression uses gradient-based optimization.
- Support vector machines use constrained optimization and duality.
- PCA, MDS, and LLE use eigendecompositions.

The optimization method follows from the modeling objective.

Evaluation

A reliable experiment separates fitting, model selection, and final testing.

- **Training set:** fit scaling, dimensionality reduction, and classifier parameters.
- **Validation set:** select kernel settings, neighbors, and embedding dimension.
- **Test set:** assess the chosen pipeline once on unseen data.

For project reports, support each conclusion with a baseline, an appropriate metric, and representative failure cases.

Connections

Different machine-learning methods make different assumptions about the data, so they optimize different objectives.

- A clustering method defines which observations belong together.
- A classifier learns a decision rule from labels.
- A dimensionality-reduction method chooses what geometric information to retain.

The useful model is the one whose assumptions, objective, and evaluation fit the actual problem.

Thank you

Questions and discussion