Locally Linear Embedding

Machine Learning

IIIy CSE IDP and IDDMP

Refer : Stephen Marsland

Motivation

- **Limitations of Linear Methods:**
  - Techniques like PCA assume a global linear structure and can fail when data lie on a curved manifold.
  - Linear projections may distort neighborhood relationships in intrinsically nonlinear datasets.
- **Preserving Local Geometry:**
  - Many real-world datasets (e.g., images, speech) live on low-dimensional curved manifolds embedded in high-dimensional space.
  - Maintaining locality (neighborhood relations) is crucial for capturing the true underlying structure.
- **Goal:**
  - Develop a non-linear dimensionality reduction technique that preserves local relationships while remaining computationally tractable.

What Is Locally Linear Embedding?

- Conceptual Overview:
  - LLE is a non-linear embedding technique
  - It seeks a low-dimensional representation that best preserves local linear reconstructions.
- Key Steps:
  - **Identify Neighbors:** For each data point, find its nearest neighbors in the original space.
  - **Compute Reconstruction Weights:** Express each point as a weighted linear combination of its neighbors, minimizing reconstruction error.
  - **Embed Globally:** Find low-dimensional points that preserve these weights, solving a single sparse eigenvalue problem.
- Outputs:
  - A set of low-dimensional coordinates where points with similar local neighborhoods remain close together

Reconstruction Error

Reconstruction Error:

- Measures how well each point x_i is approximated by a weighted sum of its neighbors.
- Defined as the sum of squared differences between the original point and its reconstruction:

$$\epsilon = \sum_{i=1}^{N} \left(\mathbf{x}_i - \sum_{j=1}^{N} \mathbf{W}_{ij} \mathbf{x}_j \right)^2 .$$

- The weights W_{ij} say how much effect the *jth* datapoint has on the reconstruction of the *ith* one.

Choosing Neighborhoods

- Only nearby points contribute meaningful information to reconstruct a given data point.
- Two common approaches:

1) Distance Threshold (Radius 'd'):

All points within a fixed distance 'd' of x_i are considered neighbors.

Neighborhood size varies depending on local density.

2) Fixed k-Nearest Neighbors:

- Exactly k closest points to x_i are chosen as neighbors.
- Guarantees constant neighborhood size, though distances may vary.

Solving for Reconstruction Weights

- **Least-Squares Formulation:**
  - For each data point x_i , we solve a constrained least-squares problem to find weights W_{ij} that minimize the reconstruction error

$$\sum_i \left\| x_i - \sum_j W_{ij}\, x_j \right\|^2$$

- **Sparsity via Neighborhoods**
  - Enforce $W_{ij}=0$ whenever x_j is not among x_i's chosen neighbors
  - This keeps the reconstruction local and the weight matrix sparse
- **Affine Constraint**
  - Impose $\sum_j \mathbf{W}_{ij} = 1$, for each i
  - Ensures invariance to translations of the data

Low-Dimensional Embedding

- **Embedding Cost Function**
  - Apply the same reconstruction idea to low-dimensional points y_i:

$$\mathbf{y}_i = \sum_{i=1}^{N} \left(\mathbf{y}_i - \sum_{j=1}^{L} \mathbf{W}_{ij} \mathbf{y}_j \right)^2$$

  - Minimizing this leads to a global eigenproblem rather than separate least-squares fits
- **Eigenproblem Construction**
  - Define matrix

$$M_{ij} = \delta_{ij} - W_{ij} - W_{ji} + \sum_k W_{ki} W_{kj}$$

    where δ_{ij} is 1 if $i = j$, else 0
  - The optimal embedding coordinates are the eigenvectors corresponding to the smallest nonzero eigenvalues of **M**

Locally Linear Embedding Algorithm

The Locally Linear Embedding Algorithm

- Decide on the neighbours of each point (e.g., K nearest neighbours):
  - compute distances between every pair of points
  - find the k smallest distances
  - set $\mathbf{W}_{ij} = 0$ for other points
  - for each point $\mathbf{x}_i$:
    - create a list of its neighbours' locations $\mathbf{z}_i$
    - compute $\mathbf{z}_i = \mathbf{z}_i - \mathbf{x}_i$
- Compute the weights matrix $\mathbf{W}$ that minimises Equation (6.31) according to the constraints:
  - compute local covariance $\mathbf{C} = \mathbf{Z}\mathbf{Z}^T$, where $\mathbf{Z}$ is the matrix of $\mathbf{z}_i$s
  - solve $\mathbf{C}\mathbf{W} = \mathbf{I}$ for $\mathbf{W}$, where $\mathbf{I}$ is the $N \times N$ identity matrix
  - set $\mathbf{W}_{ij} = 0$ for non-neighbours
  - set other elements to $\mathbf{W}/\sum(\mathbf{W})$
- Compute the lower dimensional vectors $\mathbf{y}_i$ that minimise Equation (6.32):
  - create $\mathbf{M} = (\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W})$
  - compute the eigenvalues and eigenvectors of $\mathbf{M}$
  - sort the eigenvectors into order by size of eigenvalue
  - set the qth row of $\mathbf{y}$ to be the $q+1$ eigenvector corresponding to the qth smallest eigenvalue (ignore the first eigenvector, which has eigenvalue 0)

LLE in Action – Iris & Swiss Roll

- **Iris Dataset Result**
  - When applied to Fisher's Iris data (150 samples, 4 features), LLE collapses each species into tight, well-separated clusters.
  - In fact, it effectively reduces the three species to three distinct points in the embedding space (Figure 1 in slide 10)
  - Demonstrates LLE's power to reveal clear class structure when it exists.
- **Swiss Roll Unfolding**
  - The "Swiss roll" is a 2D sheet rolled into 3D – a classic test for nonlinear DR.
  - Standard linear methods fail to flatten it without cutting, but LLE "unrolls" the manifold smoothly (Figure 2 in slide 11)
  - Highlights LLE's ability to preserve local neighborhoods and recover underlying geometry.

Iris Dataset Result

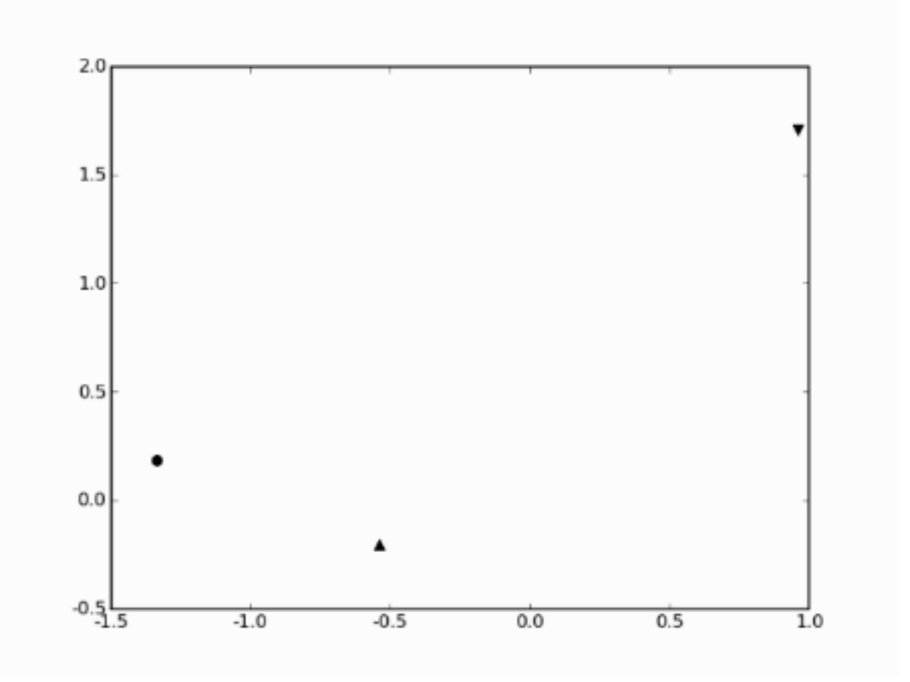

Figure 1 The Locally Linear Embedding algorithm with $k = 12$ neighbours transforms the iris dataset into three points, separating the data perfectly.

Swiss Roll Unfolding

Figure 2 A common example used to demonstrate LLE is the swissroll dataset shown on the left. To produce a useful 2D representation of this data requires unrolling the data, which the LLE does successfully, as is shown on the right. The shades are used to identify neighbouring points, and do not have any other purpose.

Summary

- Core Idea
  - Approximate a nonlinear manifold by stitching together locally linear patches.
  - Preserve neighborhood relationships rather than global distances.
- Key Steps
  - **Neighbor Selection:** For each point, choose nearby points (radius-based or k -NN).
  - **Weight Computation:** Solve a sparse, constrained least-squares problem to express each point as an affine combination of its neighbors.
  - **Embedding via Eigenproblem:** Construct matrix **M** from the weights and compute its smallest non-zero eigenvectors to obtain low-dimensional coordinates.
- Limitations
  - Sensitive to choice of neighborhood size: **too small** → disconnected patches; **too large** → violates local linearity.
  - Does not produce an explicit mapping for new points (out-of-sample extension requires additional methods).