1. Introduction

- Roughly speaking, a manifold is a locally Euclidean space.
- In 2000, two novel nonlinear manifold learning methods, Isometric feature map (Isomap) and Local linear embedding (LLE), were published in Science.
- Linear methods
  - Principal components analysis (PCA)
  - Multidimensional scaling (MDS)
- Nonlinear methods
  - Isometric feature map (Isomap)
  - Locally linear embedding (LLE)
  - Local tangent space alignment (LTSA)
  - Conformal eigenmaps
  - Laplacian eigenmaps
  - Hessian locally linear embedding (HLLE)
  - Diffusion maps
  - (Local) Riemannian manifold learning (LRML)
  - t-distributed stochastic neighbor embedding (t-SNE)
  - Uniform manifold approximation and projection (UMAP)
  - Many more variants

2. Linear manifold learning

2.1. **Principal components analysis (PCA).** (Pearson, 1901)

Assume the data $x_1, \cdots, x_n \in \mathbb{R}^q$ are centered, that is, $\sum_{i=1}^n x_{ij} = 0$ for all $j = 1, \cdots, q$. PCA computes the eigenvalues and corresponding eigenvectors of the $(q \times q)$ sample covariance matrix $S = \frac{1}{n-1} XX^T$ where $X = (x_1, \cdots, x_n)$ be the $q \times n$ data matrix. Let $X_{q\times n} = U_{q\times n} D_{n\times n} V^T_{n\times n}$ be a singular value decomposition of X.

$$\begin{aligned} S &= \frac{1}{n-1} UD^2U^T \qquad (1) \\ &= \frac{1}{n-1} U\Lambda U^T \qquad (2) \end{aligned}$$

where $\lambda_1 \geq \cdots \geq \lambda_q \geq 0$ and $\Lambda = D^2$.

$$\begin{aligned} Y &= U^T X = DV^T \qquad (3) \\ Y^T &= (d_1 v_1 : \cdots : d_q v_q) \qquad (4) \end{aligned}$$

For a k-dimensional embedding for $p < q$, PCA retains the first p eigenvectors that explain a high percentage of the total variation (sum of the eigenvalues $\sum_{j=1}^p \lambda_j$) in the data.

$$(y_1, \cdots, y_p) = \left(\sqrt{\lambda_1} v_1 : \cdots : \sqrt{\lambda_p} v_p\right) \qquad (5)$$

since $d_j = \pm\sqrt{\lambda_j}$ and ignoring orientations. Roughly speaking, PCA is an eigen-problem of XX^T.

2.2. **Multidimensional scaling (MDS).** MDS was first proposed in (Torgerson, 1952).

Let $D_2 = (d_{ij}^2) = (d_X^2(x_i.x_j))$ be the matrix of the squared Euclidean distances. Let $H = I = \frac{1}{n} 11^T$ be the centering matrix.

2.2.1. *Classical MDS.* The relationship between Gram matrix and the squared Euclidean distance matrix:

$$X^T X = -\frac{1}{2} H D_2 H \tag{6}$$

The coordinates can be recovered from the pairwise distances. The classical MDS tries to find the embedding that preserving the similarities or distances of the data as much as possible by minimizing

$$\min_{Y} \left\| X^T X - Y^T Y \right\|^2 \tag{7}$$

For a p-dimensional embedding for $p < q$, it can be shown that the solution of the classical MDS is the same as that of PCA.

$$(Y^T)_{n\times p} = V_{n\times p} \Lambda_{p\times p}^{1/2} \tag{8}$$

2.2.2. *Metric MDS.* The classical MDS tries to preserve the similarities (inner products) of points in the embedding space. Metric MDS tries to preserve the distances of points in the embedding space. It minimizes the difference of distances of points in the input and embedding spaces. The cost function in metric MDS is usually referred to as the stress function. This method is named metric MDS because it uses distance metric in its optimization. The optimization in metric MDS is:

$$\left(\frac{\sum_{i=1}^{n} \sum_{j=1, j<i}^{n} \left(d_x(x_i, x_j) - d_y(y_i, y_j) \right)^2}{\sum_{i=1}^{n} \sum_{j=1, j<i}^{n} d_x^2(x_i, x_j)} \right)^{\frac{1}{2}} \tag{9}$$

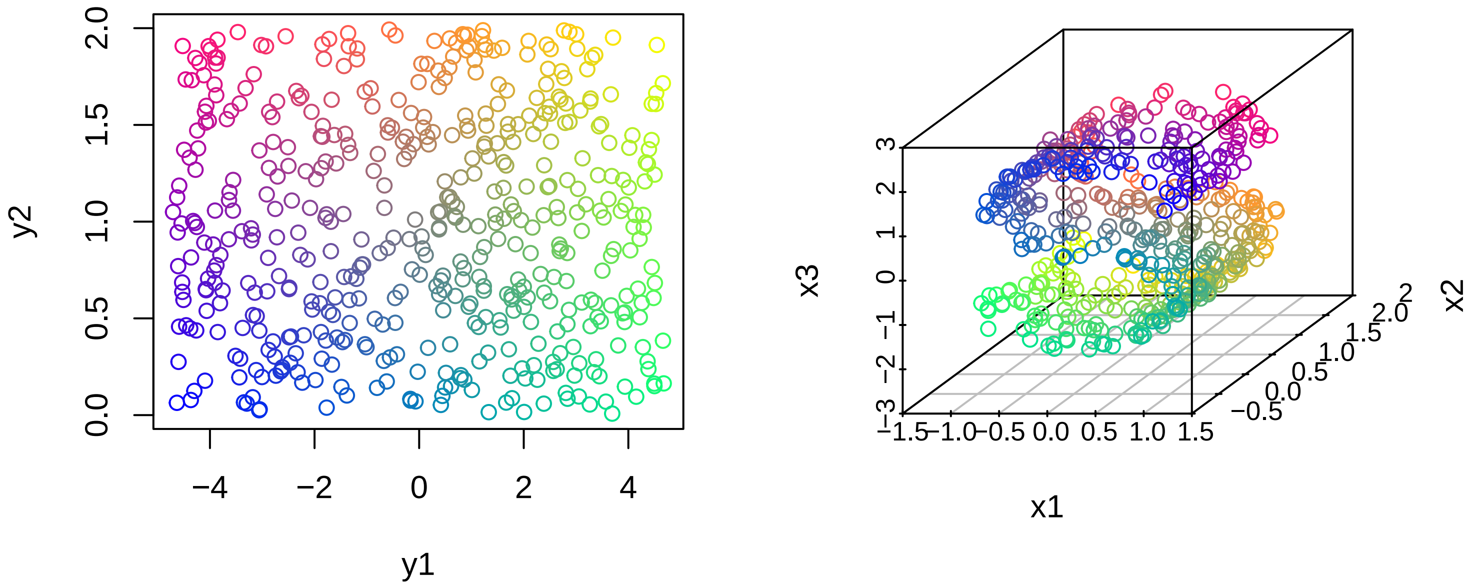
or

$$\left(\sum_{i=1}^{n} \sum_{j=1, j<i}^{n} \left(d_x(x_i, x_j) - d_y(y_i, y_j) \right)^2 \right)^{\frac{1}{2}} \tag{10}$$

2.2.3. *Example.* PCA and MDS for Swiss roll data.

In Swiss roll data, 2-dimensional data points are mapped in 3-dimensional space by an embedding $\phi : \mathbb{R}^2 \to \mathbb{R}^3$ where $\phi(y_1, y_2) = (x_1, x_2, x_3)$. The embedding is a nonlinear function. Suppose Swiss roll data in 3-dimensional space are given. Can we unfold Swiss roll by finding the inverse map ϕ^{-1} from $\mathbb{R}^3$ to $\mathbb{R}^2$?

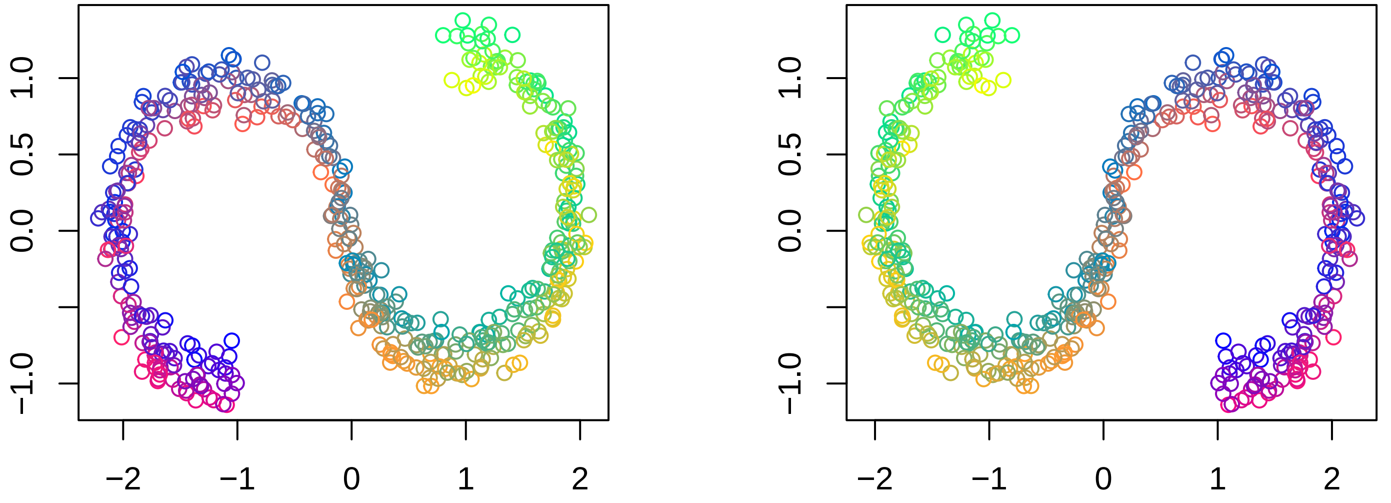
Figure 1. Swiss roll data



Apply the two linear dimensionality reduction methods, PCA and MDS, to see if they can unfold nonlinear embedding.

```
par(mfrow = c(1, 2))
emb <- embed(dat, "PCA")
plot(emb, type = "2vars", xlab = "", ylab = "")

emb <- embed(dat, "MDS")
plot(emb, type = "2vars", xlab = "", ylab = "")
```

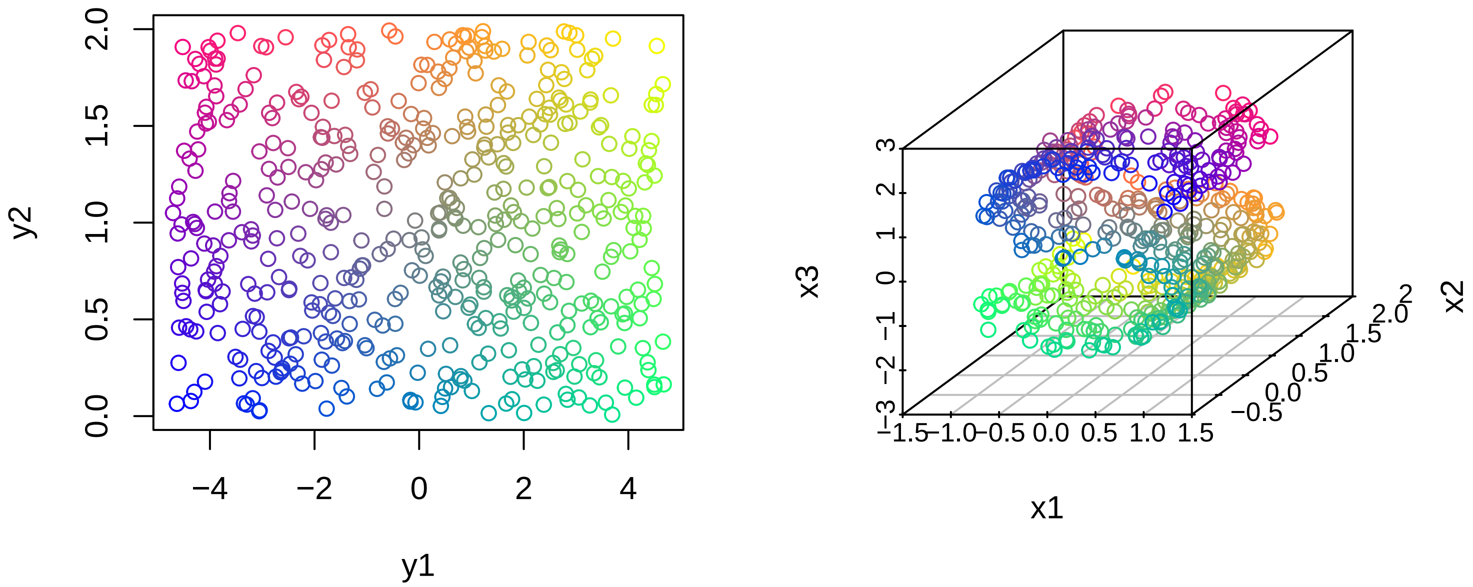
FIGURE 2. PCA and MDS for Swiss roll data



It usually consists of three steps:

1. A nearest neighbor search
2. Distances or dissimilarities
3. Eigen problem

Figure 3. Swiss roll data



3.1. **Isometric feature map (Isomap).** Isomap (Tenenbaum et al., 2000) is a special case of the generalized classical MDS. Rather than the Euclidean distance, Isomap uses an approximation of the geodesic distance. As was explained, the classical MDS is linear; hence, it cannot capture the nonlinearity of the manifold. Isomap makes use of the geodesic distance to make the generalized classical MDS nonlinear.

Assume the data set $\mathcal{X} = \{x_i : i = 1, \cdots, n\} \subset M \subset \mathbb{R}^q$ and f is an isometry from M to $\mathbb{R}^q$ such that $f(x_i) = y_i$, where M is a p-dimensional Riemannian manifold.

(1) Define the neighborhood of a data point by k-nearest neighbors (k-isomap) or ϵ-distance neighbors (ϵ-isomap).

(2) Create a graph $[\mathcal{X}, E]$ and compute graph distances.

$$S_G = \left(d_G^2(x_i, x_j)\right) \tag{11}$$

(3) Construct the isomap kernel

$$K = -\frac{1}{2} H S_G H \tag{12}$$

(4) Eigen decomposition to obtain $Y = (y_1, \cdots, y_n)$

$$K = V \Lambda V^T \tag{13}$$

$$Y^T = \left(\sqrt{\lambda_1} v_1, \cdots, \sqrt{\lambda_p} v_p\right) \tag{14}$$

3.2. **Locally linear embedding (LLE).** (Roweis & Saul, 2000) As in Isomap, knn is applied to construct a neighbor graph $G = (V, E)$. A weight matrix is calculated by minimizing

$$\sum_{i=1}^{n} \|x_i - \sum_j W_{ij} x_j\|^2 \tag{15}$$

subject to $\sum_{j-1}^{n} W_{ij} = 1$ and $W_{ij} = 0$ if $x_j \notin N(x_i)$. The weight matrix W can be simply obtained by LSE. Once W is obtained, we find y_i that minimizes

$$\sum_{i=1}^{n} \left\| y_i - \sum_{j} W_{ij} y_j \right\|^2 \tag{16}$$

subject to $\sum_{i=1}^{n} y_i = 0$ and $\frac{1}{n} \sum_{i=1}^{n} y_i y_i^T = I$. Let $K = (I - W)^T (I - W)$ be a sparse symmetric matrix. Let $Y = (y_1, \cdots, y_n)$ be a $p \times n$ matrix.

$$\sum_{i=1}^{n} \left\| y_i - \sum_{j} W_{ij} y_j \right\|^2 = tr\left(YKY^T\right) \tag{17}$$

The Lagrangian function can be written as

$$\mathcal{L} = tr\left(YKY^T\right) - tr\left(\Lambda^T \left(\frac{1}{n} YY^T - I\right)\right) \tag{18}$$

The solution is equivalent to an eigenvalue-eigenvector problem.

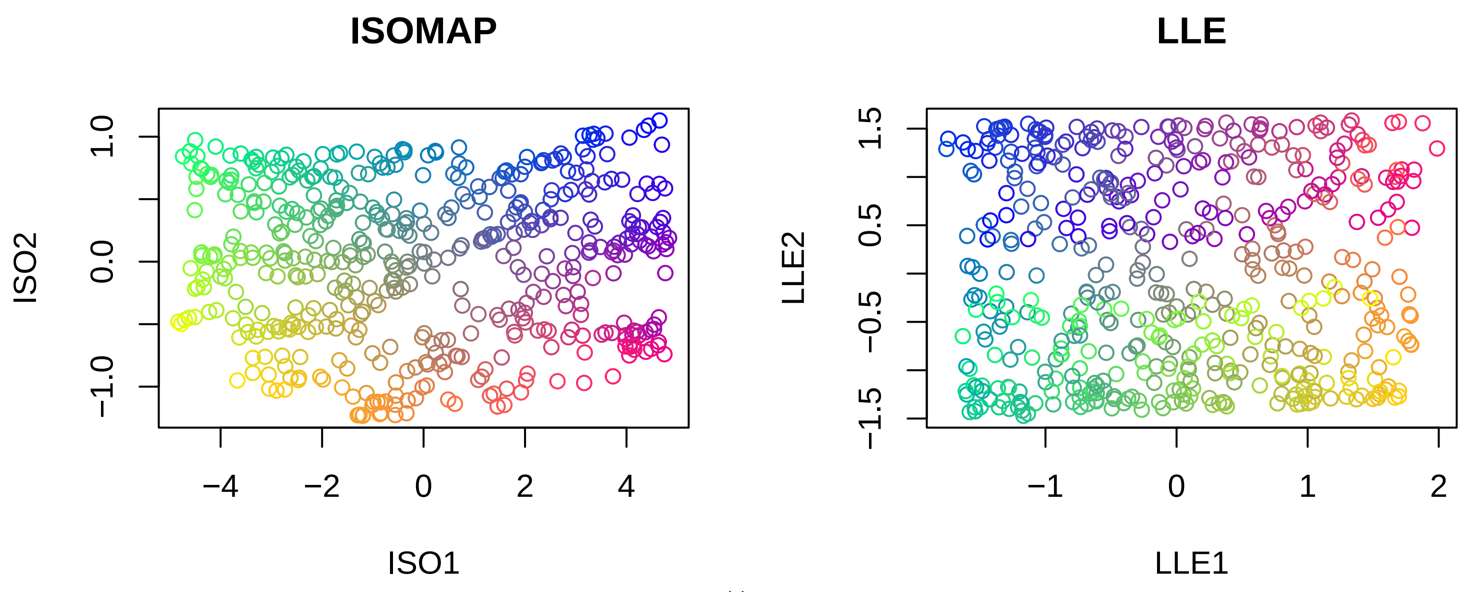
$$KY^T = \frac{1}{n} Y^T \Lambda \tag{19}$$

Note the smallest eigenvalue of K is equal to 0 with the eigenvector 1. We select the smallest $(p+1)$ eigenvalues (0 is included) and the correspoding eigenvectors produce Y without 1, that is, $Y^T = (e_1, \cdots, e_p)$ where $Ke_i = d_i e_i$ and $d_0 = 0 < d_1 \leq \cdots \leq d_p \leq \cdots \leq d_{n-1}$.

3.2.1. *Example.* Swiss roll with Isomap and LLE

```
library(dimRed)
set.seed(1000)
dat <- loadDataSet("3D S Curve", n = 500)
par(mfrow = c(1, 2))
emb <- embed(dat, "Isomap", knn = 10)
plot(emb, type = "2vars", xlab = "ISO1", ylab = "ISO2", main = "ISOMAP")
emb <- embed(dat, "LLE", knn = 45)
plot(emb, type = "2vars", xlab = "LLE1", ylab = "LLE2", main = "LLE")
```

FIGURE 4. Isomap and LLE for Swiss roll data



3.3. **Locally Tangent Space Alignment (LTSA).** (Zhang & Zha, 2004)

(1) Define the neighborhood of a data point by k-nearest neighborhood or ϵ-distance neighborhood.

(2) Locally apply PCA to construct local coordinates

$$X_i H = U_i \Sigma_i V_i^T \tag{20}$$

where X_i is a $q \times k$ matrix, H is the centering matrix, U_i, Σ_i, and V_i are the reduced matrices for $p-$dimensional embedding, i.e., U_i is a $q \times p$ matrix, Σ_i is a $p \times p$ matrix, and V_i is a $k \times p$ matrix. Let $G_i = \begin{pmatrix} \frac{1}{\sqrt{k}} 1 & V_i \end{pmatrix}$.

$$W_i = I - G_i G_i^T \tag{21}$$

(3) Construct a global kernel

Let S_i be the $n \times k$ selection matrix such that $X_i = XS_i$ and $Y_i = YS_i$. Let $S = (S_1, \cdots, S_n)$ and $W = \mathrm{diag}(W_1, \cdots, W_n)$.

$$K = SWW^T S^T \tag{22}$$

(4) Solve the eigen problem Let $e_1, \cdots, e_p$ be the corresponding eigenvectors of K such that the eigenvalues are $d_0 = 0 < d_1 \leq \cdots \leq d_p \leq \cdots \leq d_{n-1}$.

$$Y^T = \begin{pmatrix} e_1 & \cdots & e_p \end{pmatrix} \tag{23}$$

3.4. **Laplacian eigenmap.** (Belkin & Niyogi, 2001)

(1) Define the neighborhood of a data point by k-nearest neighborhood or ϵ-distance neighborhood.

(2) Construct the weighted graph

$$w_{ij} = \begin{cases} \exp\left(-\frac{||x_i - x_j||^2}{4t}\right) & \text{for } j \in N(x_i) \\ 0 & \text{o.w.} \end{cases} \tag{24}$$

$$d_i = \sum_{j \in N(x_i)} \exp\left(-\frac{||x_i - x_j||^2}{4t}\right) \tag{25}$$

(3) Construct the discrete Laplacian kernel

$$L = D - W \tag{26}$$

$$\mathcal{L} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2} \tag{27}$$

(4) Eigen decomposition

$$Y = \text{argmin}\left\{\text{tr}\left(Y \mathcal{L} Y^T\right)\right\} \tag{28}$$

Let e_i be the eigenvector of $\mathcal{L}$ such that $0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_p \leq \cdots \leq \lambda_{n-1}$.

$$Y^T = (e_1, \cdots, e_p) \tag{29}$$

3.5. **t-distributed stochastic neighbor embedding (t-SNE).** (Hinton & Roweis, 2002; Van der Maaten & Hinton, 2008)

The t-SNE is a nonlinear dimensionality reduction method well-suited for embedding high-dimensional data for visualization in a low-dimensional space of 2 or 3 dimensions. Similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability.

(1) Compute the similarity of two points proportional to the probability p_{ij}

$$p_{j|i} = \frac{\exp\left(-||x_i - x_j||^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-||x_i - x_k||^2/2\sigma_i^2\right)} \tag{30}$$

$$p_{i|i} = 0, \sum_j p_{j|i} = 1 \tag{31}$$

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{n} \tag{32}$$

Note $p_{ij} = p_{ji}, p_{ii} = 0, \sum_{i,j} p_{ij} = 1$.

(2) Define q_{ij} the similarity of y_i and y_j in a low dimensional space

$$q_{ij} = \frac{\left(1 + ||y_i - y_j||^2\right)^{-1}}{\sum_k \sum_{l\neq k}\left(1 + ||y_k - y_l||^2\right)^{-1}} \tag{33}$$

$$q_{ii} = 0 \tag{34}$$

(3) Minimize Kullback-Leibler divergence of P from Q

$$KL(P||Q) = \sum_{i\neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{35}$$

3.6. **Uniform manifold approximation and projection (UMAP).** (McInnes et al., 2018)
The UMAP algorithm is developed under three assumptions on the data

(1) The data is uniformly distributed on a Riemannian manifold

(2) The Riemannian metric is (approximately) locally constant

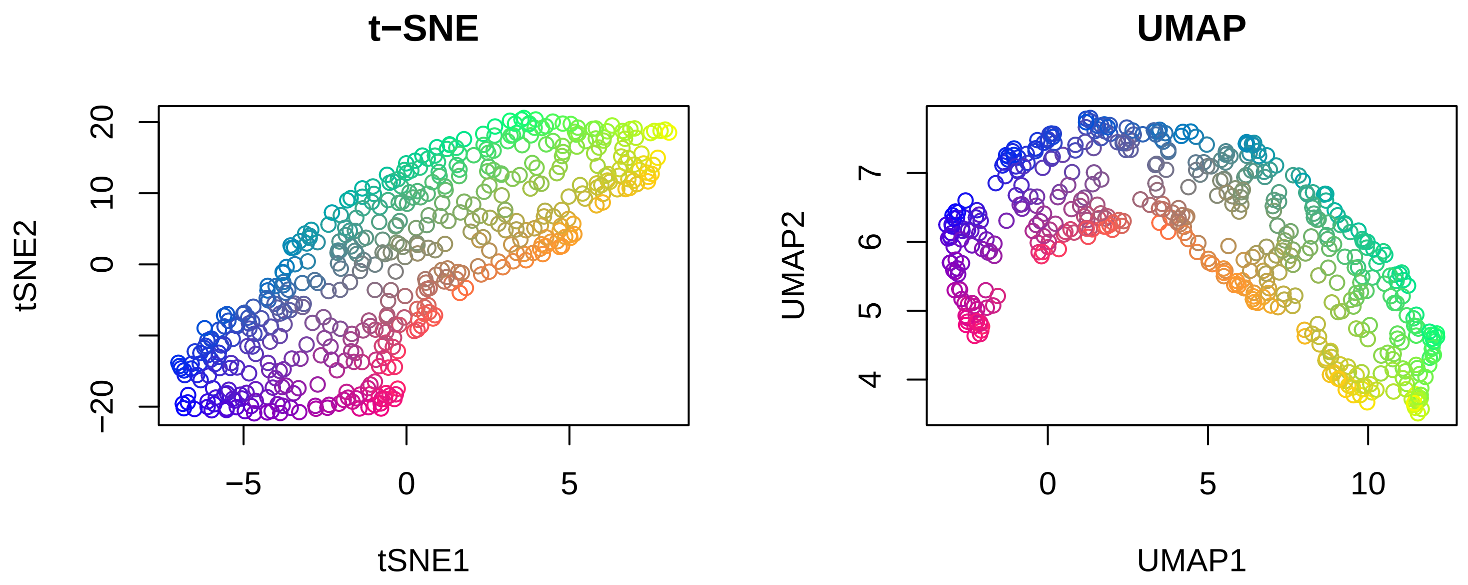
(3) The manifold is locally connected

Based on these assumptions, it is possible to model the manifold with a fuzzy topological structure. The embedding is constructed by searching for a low dimensional (usually 2 or 3 dimensional) projection of the data that has the closest possible equivalent fuzzy topological structure.

3.6.1. *Example.* Swiss roll with t-SNE and UMAP

```
library(dimRed)
set.seed(1000)
dat <- loadDataSet("3D S Curve", n = 500)
par(mfrow = c(1, 2))
emb <- embed(dat, "tSNE", perplexity = 50)
plot(emb, type = "2vars", xlab = "tSNE1", ylab = "tSNE2", main = "t-SNE")
emb <- embed(dat, "UMAP", knn = 100)
plot(emb, type = "2vars", xlab = "UMAP1", ylab = "UMAP2", main = "UMAP")
```

FIGURE 5. t-SNE and UMAP for Swiss roll data

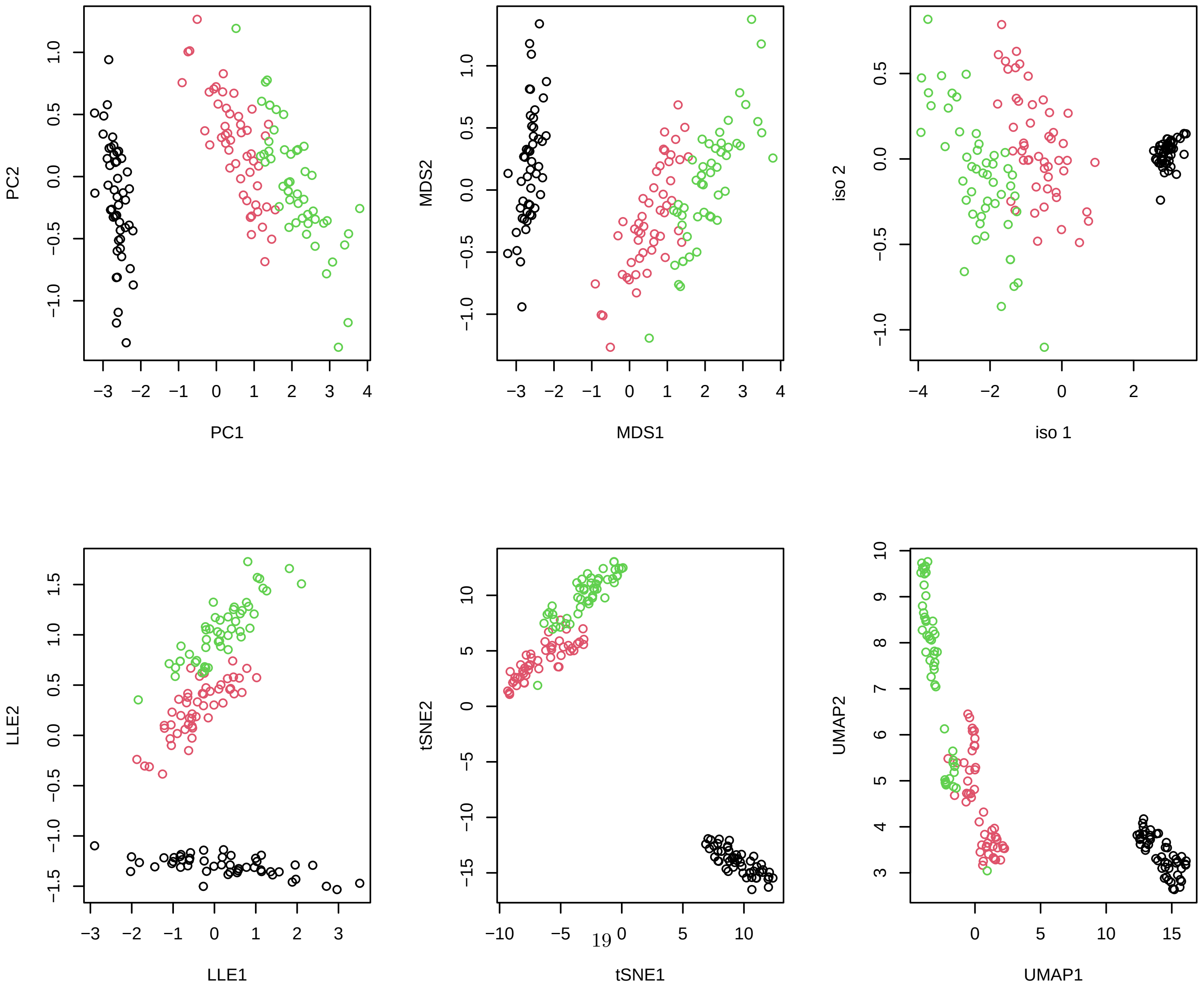


3.6.2. *Example.* iris data

```r
par(mfrow = c(2, 3))
library(dimRed)
dat = iris[, -5]

emb <- embed(dat, "PCA")
plot(emb@data@data, col = as.integer(iris[, 5]))

emb <- embed(dat, "MDS")
plot(emb@data@data, col = as.integer(iris[, 5]))

emb <- embed(dat, "Isomap")
plot(emb@data@data, col = as.integer(iris[, 5]))

emb <- embed(dat, "LLE")
plot(emb@data@data, col = as.integer(iris[, 5]))

emb <- embed(dat, "tSNE")
plot(emb@data@data, col = as.integer(iris[, 5]))
```

```r
emb <- embed(dat, "UMAP")
plot(emb@data@data, col = as.integer(iris[, 5]))
```

Figure 6. PCA, MDS, Isomap, LLE, t-SNE and UMAP for iris data

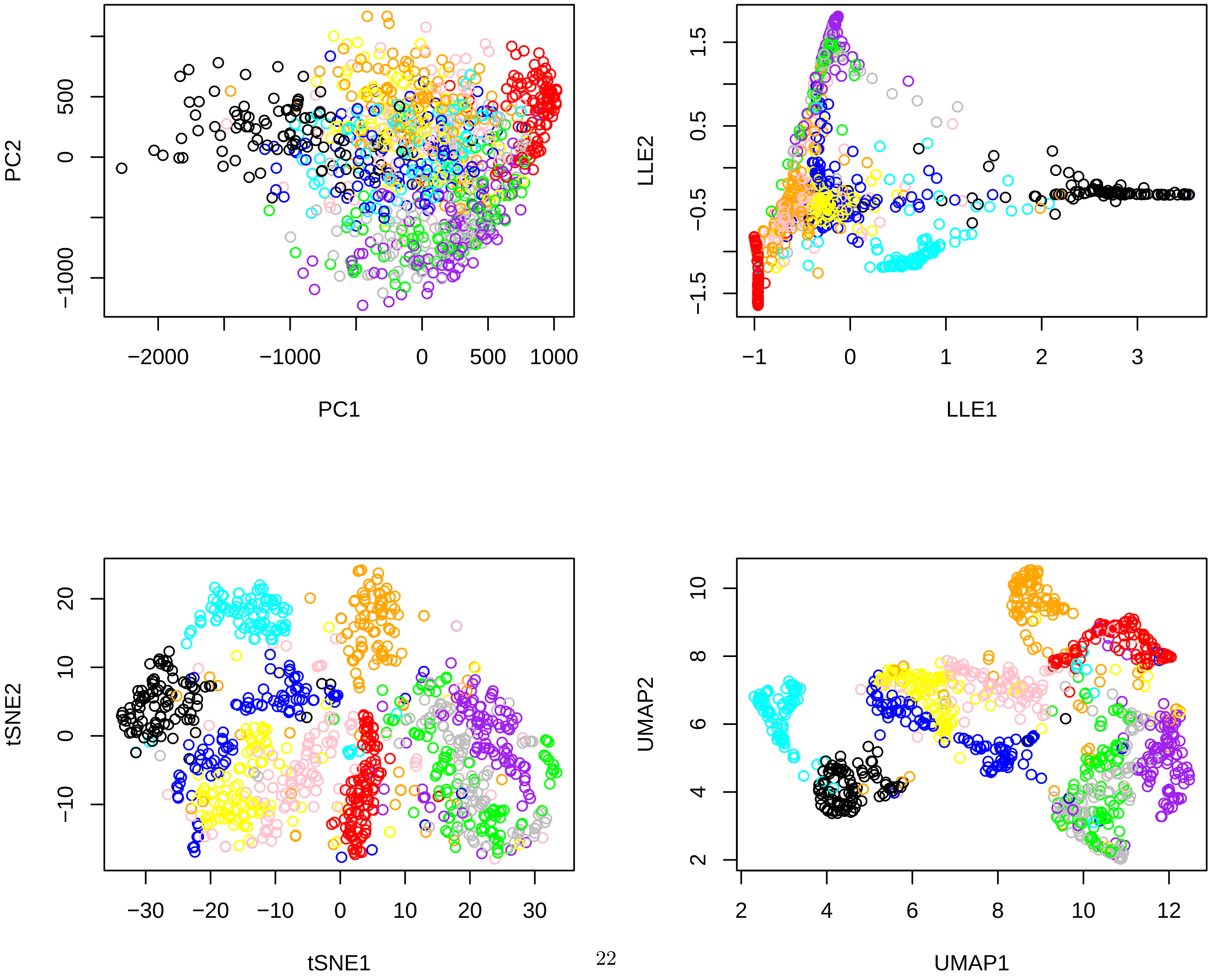


3.6.3. *Example.* Visualizing a subset of MNIST data.

```r
library(keras)
mnist <- dataset_mnist()
x_train <- mnist$train$x
y_train <- mnist$train$y
x_train <- array_reshape(x_train, c(nrow(x_train), 784))
library(caret)
set.seed(1)
train <- createDataPartition(y_train, p = 1/60, list = F)
x_train <- x_train[train, ]
y_train <- y_train[train]
dat <- data.frame(x_train)
pp <- preProcess(dat, method = c("zv"))
dat <- predict(pp, dat)

par(mfrow = c(2, 2))
library(dimRed)
col = c("black", "red", "orange", "yellow", "green", "blue", "cyan",
    "purple", "pink", "gray")
emb <- embed(dat, "PCA")
plot(emb@data@data, col = col[y_train + 1])
```

```r
emb <- embed(dat, "LLE", knn = 5)
plot(emb@data@data, col = col[y_train + 1])

emb <- embed(dat, "tSNE")
plot(emb@data@data, col = col[y_train + 1])

emb <- embed(dat, "UMAP")
plot(emb@data@data, col = col[y_train + 1])
```

FIGURE 7. 1,000 observations of MNIST data


PC2
PC1
LLE2
LLE1
tSNE2
tSNE1
UMAP2
UMAP1