Topology Preservation II: The Lattice Strikes Back

Yoan Miche

CIS, HUT

November 6, 2007

What's new?

Previously. . .

- Topology lattice predefined
- Based on some "intuition"/idea for the data
- In the best case scenario:
  - You know the data topology
  - You select the nicest lattice shape to preserve it

But life is cruel

Usually no idea beforehand about data topology (or not much)

What's new?

Previously...

- Topology lattice predefined
- Based on some "intuition"/idea for the data
- In the best case scenario:
  - You know the data topology
  - You select the nicest lattice shape to preserve it

But life is cruel

Usually no idea beforehand about data topology (or not much)

What's new?

So, what about trying to infer a topology from the data itself?

Ideally, then:

- Unconstrained embedding
- More adaptive

Now how do we do this?

Lattice is in fact a graph built from the data:

- Vertices: data points
- Edges: neighbourhood relationships

What's new?

So, what about trying to infer a topology from the data itself?
Ideally, then:

- Unconstrained embedding
- More adaptive

Now how do we do this?

Lattice is in fact a graph built from the data:

- Vertices: data points
- Edges: neighbourhood relationships

Outline

1. Creating these graphs
2. Locally Linear Embedding (LLE)
3. Laplacian Eigenmap (LE)

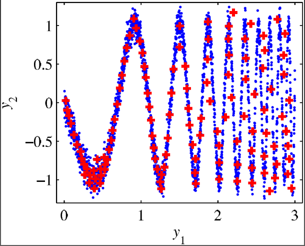
As seen before...

We just want to connect neighbouring points of the space.

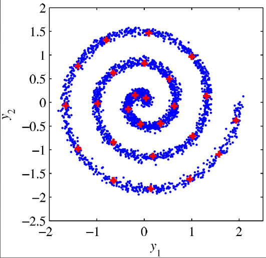
Two main situations:

- Data not quantized
- Data quantized

And two color figures to divert you...



(a) Sine example

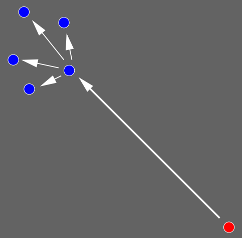


(b) Spiral example

K-rule - Does not exactly work well on the examples...

The K-rule

- Find the K closest points

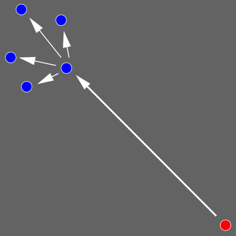


- Choosing incorrectly K may lead to “wrong” neighbours
- And hence, edges issues

K-rule - Does not exactly work well on the examples. . .

The K-rule

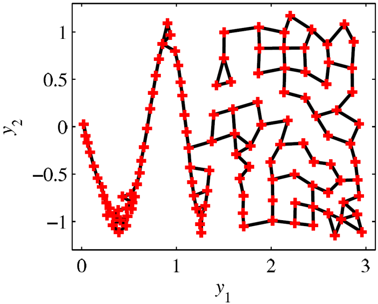
- Find the K closest points



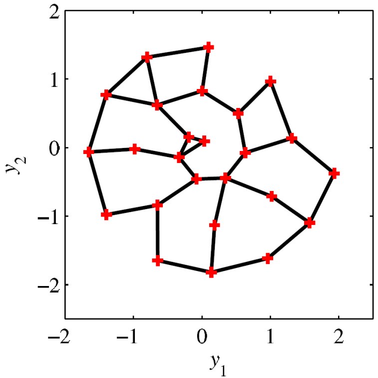
- Choosing incorrectly K may lead to “wrong” neighbours
- And hence, edges issues

K-rule - Does not exactly work well on the examples. . . (2)

Edges and neighbours issues, for example



(c) Sine example



(d) Spiral example

ϵ-rule - Funny results also...

ϵ-rule

- Each point gets connected to points within an ϵ radius ball (centered on the considered point)
- But then... Isolated points may have no neighbours (or "wrong" ones)
- Hard to evaluate a proper ϵ (more than K) in practice
- Results are OK when data is uniformly distributed
  - Too dense $\Longrightarrow$ Too many edges
  - Too sparse $\Longrightarrow$ Disconnected points

ϵ-rule - Funny results also...

ϵ-rule

- Each point gets connected to points within an ϵ radius ball (centered on the considered point)
- But then... Isolated points may have no neighbours (or "wrong" ones)
- Hard to evaluate a proper ϵ (more than K) in practice
- Results are OK when data is uniformly distributed
  - Too dense $\Longrightarrow$ Too many edges
  - Too sparse $\Longrightarrow$ Disconnected points

ϵ-rule - Funny results also...

ϵ-rule

- Each point gets connected to points within an ϵ radius ball (centered on the considered point)

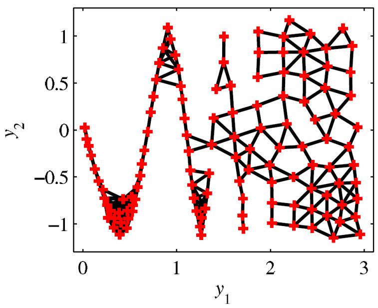
- But then... Isolated points may have no neighbours (or "wrong" ones)
- Hard to evaluate a proper ϵ (more than K) in practice
- Results are OK when data is uniformly distributed
  - Too dense $\Longrightarrow$ Too many edges
  - Too sparse $\Longrightarrow$ Disconnected points

ϵ-rule - Funny results also. . .

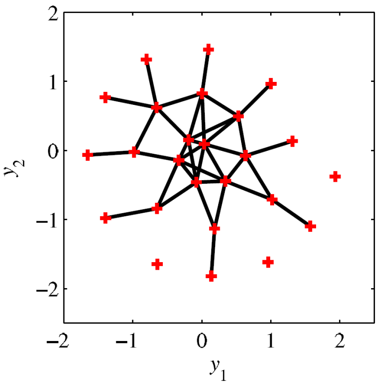
ϵ-rule

- Each point gets connected to points within an ϵ radius ball (centered on the considered point)
- But then. . . Isolated points may have no neighbours (or “wrong” ones)
- Hard to evaluate a proper ϵ (more than K) in practice
- Results are OK when data is uniformly distributed
  - Too dense $\Longrightarrow$ Too many edges
  - Too sparse $\Longrightarrow$ Disconnected points

ϵ-rule - Funny results also... (2)



(e) Sine example



(f) Spiral example

τ-rule - Slightly better...

τ-rule

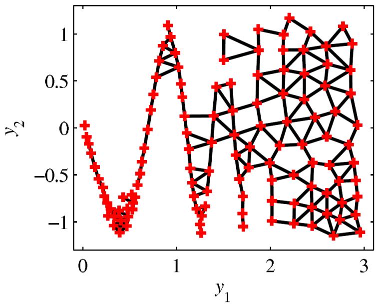
Two points $\mathbf{y}(i)$ and $\mathbf{y}(j)$ are connected if

$$\overbrace{\min_j ||\mathbf{y}(i) - \mathbf{y}(j)||}^{d_i} \leq \tau \overbrace{\min_i ||\mathbf{y}(j) - \mathbf{y}(i)||}^{d_j} \text{ and } d_j \leq \tau d_i (\text{similarity cond.}) \tag{1}$$

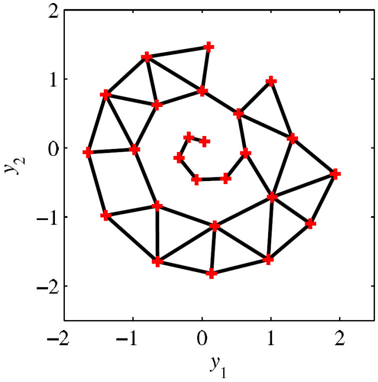
$$||\mathbf{y}(i) - \mathbf{y}(j)|| \leq \tau d_i \text{ or } ||\mathbf{y}(j) - \mathbf{y}(i)|| \leq \tau d_j (\text{neighborhood cond.}) \tag{2}$$

Behaves almost like the ϵ-rule but with an implicit radius (in τ)

τ-rule - Slightly better. . . (2)



(g) Sine example



(h) Spiral example

Data rule: let's be serious

Why not use the information of the quantization for the graph building ?

Data rule

- for each point $\mathbf{y}(i)$
- Each pair $\mathbf{c}(j_s), \mathbf{c}(j_t)$ has to follow the two conditions to be connected:

Data rule: let's be serious

Why not use the information of the quantization for the graph building ?

Data rule

- for each point $\mathbf{y}(i)$
  - compute the K closest prototypes $\mathbf{c}(j_1), \ldots, \mathbf{c}(j_K)$
- Each pair $\mathbf{c}(j_s), \mathbf{c}(j_t)$ has to follow the two conditions to be connected:
  - "Condition of the ellipse":

$$d(\mathbf{y}(i), \mathbf{c}(j_s)) + d(\mathbf{y}(i), \mathbf{c}(j_t)) < C_1 d(\mathbf{c}(j_s), \mathbf{c}(j_t)) \quad (3)$$

  - "Condition of the circle":

$$d(\mathbf{y}(i), \mathbf{c}(j_s)) < C_2 d(\mathbf{y}(i), \mathbf{c}(j_t)) \text{ and } d(\mathbf{y}(i), \mathbf{c}(j_t)) < C_2 d(\mathbf{y}(i), \mathbf{c}(j_s)) \quad (4)$$

Data rule: let's be serious

Why not use the information of the quantization for the graph building ?

Data rule

- for each point $\mathbf{y}(i)$
  - compute the K closest prototypes $\mathbf{c}(j_1), \ldots, \mathbf{c}(j_K)$
- Each pair $\mathbf{c}(j_s), \mathbf{c}(j_t)$ has to follow the two conditions to be connected:
  - "Condition of the ellipse":

$$d(\mathbf{y}(i), \mathbf{c}(j_s)) + d(\mathbf{y}(i), \mathbf{c}(j_t)) < C_1 d(\mathbf{c}(j_s), \mathbf{c}(j_t)) \tag{3}$$

  - "Condition of the circle":

$$d(\mathbf{y}(i), \mathbf{c}(j_s)) < C_2 d(\mathbf{y}(i), \mathbf{c}(j_t)) \text{ and } d(\mathbf{y}(i), \mathbf{c}(j_t)) < C_2 d(\mathbf{y}(i), \mathbf{c}(j_s)) \tag{4}$$

Data rule: let's be serious

Why not use the information of the quantization for the graph building ?

Data rule

- for each point $\mathbf{y}(i)$
  - compute the K closest prototypes $\mathbf{c}(j_1), \ldots, \mathbf{c}(j_K)$
- Each pair $\mathbf{c}(j_s), \mathbf{c}(j_t)$ has to follow the two conditions to be connected:
  - "Condition of the ellipse":

$$d(\mathbf{y}(i), \mathbf{c}(j_s)) + d(\mathbf{y}(i), \mathbf{c}(j_t)) < C_1 d(\mathbf{c}(j_s), \mathbf{c}(j_t)) \quad (3)$$

  - "Condition of the circle":

$$d(\mathbf{y}(i), \mathbf{c}(j_s)) < C_2 d(\mathbf{y}(i), \mathbf{c}(j_t)) \text{ and } d(\mathbf{y}(i), \mathbf{c}(j_t)) < C_2 d(\mathbf{y}(i), \mathbf{c}(j_s)) \quad (4)$$

Data rule: let's be serious

Why not use the information of the quantization for the graph building ?

Data rule

- for each point $\mathbf{y}(i)$
  - compute the K closest prototypes $\mathbf{c}(j_1), \ldots, \mathbf{c}(j_K)$
- Each pair $\mathbf{c}(j_s), \mathbf{c}(j_t)$ has to follow the two conditions to be connected:
  - "Condition of the ellipse":

$$d(\mathbf{y}(i), \mathbf{c}(j_s)) + d(\mathbf{y}(i), \mathbf{c}(j_t)) < C_1 d(\mathbf{c}(j_s), \mathbf{c}(j_t)) \quad (3)$$

  - "Condition of the circle":

$$d(\mathbf{y}(i), \mathbf{c}(j_s)) < C_2 d(\mathbf{y}(i), \mathbf{c}(j_t)) \text{ and } d(\mathbf{y}(i), \mathbf{c}(j_t)) < C_2 d(\mathbf{y}(i), \mathbf{c}(j_s)) \quad (4)$$

Data rule: let's be serious

Why not use the information of the quantization for the graph building ?

Data rule

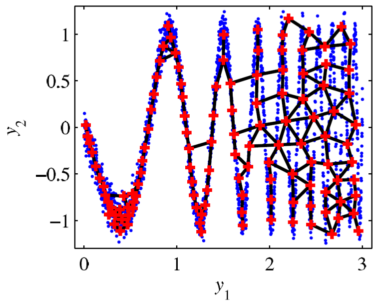
- for each point $\mathbf{y}(i)$
  - compute the K closest prototypes $\mathbf{c}(j_1), \ldots, \mathbf{c}(j_K)$
- Each pair $\mathbf{c}(j_s), \mathbf{c}(j_t)$ has to follow the two conditions to be connected:
  - "Condition of the ellipse":

$$d(\mathbf{y}(i), \mathbf{c}(j_s)) + d(\mathbf{y}(i), \mathbf{c}(j_t)) < C_1 d(\mathbf{c}(j_s), \mathbf{c}(j_t)) \quad (3)$$

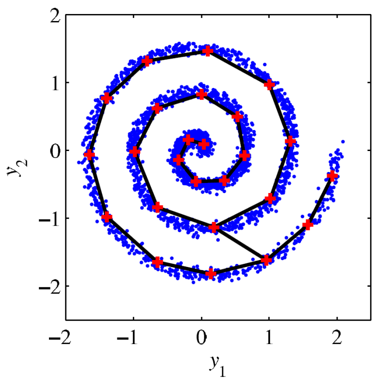
  - "Condition of the circle":

$$d(\mathbf{y}(i), \mathbf{c}(j_s)) < C_2 d(\mathbf{y}(i), \mathbf{c}(j_t)) \text{ and } d(\mathbf{y}(i), \mathbf{c}(j_t)) < C_2 d(\mathbf{y}(i), \mathbf{c}(j_s)) \quad (4)$$

Looks nicer. . .

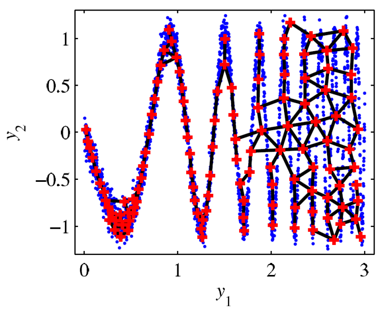


(i) Sine example

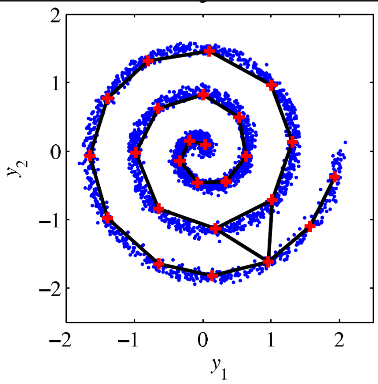


(j) Spiral example

Histogram rule exists also. . .



(k) Sine example



(l) Spiral example

Ideas behind...

- While SOM and GTM try to preserve neighbouring points close, we work on angles with LLE
- LLE uses **conformal mapping** to preserve local angles
- This is somewhat related to preserving distances: aims at preserving the local scalar product properties

Ideas behind...

- While SOM and GTM try to preserve neighbouring points close, we work on angles with LLE
- LLE uses **conformal mapping** to preserve local angles
- This is somewhat related to preserving distances: aims at preserving the local scalar product properties

Ideas behind...

- While SOM and GTM try to preserve neighbouring points close, we work on angles with LLE
- LLE uses **conformal mapping** to preserve local angles
- This is somewhat related to preserving distances: aims at preserving the local scalar product properties

A little algorithm...

LLE algorithm

1. Determine which angles to take into account
   - Select neighbours for each point using a previous graph building technique (mostly K closest or ϵ ball)
2. Then, replace each data point with a linear combination of the selected neighbours
3. Local geometry of manifold characterized by these linear coefficients.
4. Reconstruction error measured by

$$E(\mathbf{W}) = \sum_{i=1}^{N} \left|\left| \mathbf{y}(i) - \sum_{j \in \mathcal{N}(i)} w_{i,j} \mathbf{y}(j) \right|\right|^2 \tag{5}$$

With $\mathcal{N}(i)$ the set of neighbours of $\mathbf{y}(i)$ and $w_{i,j}$ the coefficients of the $N \times N$ matrix $\mathbf{W}$ of weights

A little algorithm...

LLE algorithm

1. Determine which angles to take into account
   - Select neighbours for each point using a previous graph building technique (mostly K closest or ϵ ball)
2. Then, replace each data point with a linear combination of the selected neighbours
3. Local geometry of manifold characterized by these linear coefficients.
4. Reconstruction error measured by

$$E(\mathbf{W}) = \sum_{i=1}^{N} \left|\left| \mathbf{y}(i) - \sum_{j \in \mathcal{N}(i)} w_{i,j} \mathbf{y}(j) \right|\right|^2 \qquad (5)$$

With $\mathcal{N}(i)$ the set of neighbours of $\mathbf{y}(i)$ and $w_{i,j}$ the coefficients of the $N \times N$ matrix $\mathbf{W}$ of weights

A little algorithm…

LLE algorithm

1. Determine which angles to take into account
   - Select neighbours for each point using a previous graph building technique (mostly K closest or ϵ ball)
2. Then, replace each data point with a linear combination of the selected neighbours
3. Local geometry of manifold characterized by these linear coefficients.
4. Reconstruction error measured by

$$E(\mathbf{W}) = \sum_{i=1}^{N} \left\| \mathbf{y}(i) - \sum_{j \in \mathcal{N}(i)} w_{i,j} \mathbf{y}(j) \right\|^2 \tag{5}$$

With $\mathcal{N}(i)$ the set of neighbours of $\mathbf{y}(i)$ and $w_{i,j}$ the coefficients of the $N \times N$ matrix $\mathbf{W}$ of weights

A little algorithm...

LLE algorithm

1. Determine which angles to take into account
   - Select neighbours for each point using a previous graph building technique (mostly K closest or ϵ ball)
2. Then, replace each data point with a linear combination of the selected neighbours
3. Local geometry of manifold characterized by these linear coefficients.
4. Reconstruction error measured by

$$E(\mathbf{W}) = \sum_{i=1}^{N} \left\| \mathbf{y}(i) - \sum_{j \in \mathcal{N}(i)} w_{i,j} \mathbf{y}(j) \right\|^2 \tag{5}$$

With $\mathcal{N}(i)$ the set of neighbours of $\mathbf{y}(i)$ and $w_{i,j}$ the coefficients of the $N \times N$ matrix $\mathbf{W}$ of weights

A little algorithm…(2)

To compute the $w_{i,j}$, $E(\mathbf{W})$ is minimized under two constraints

- Points are reconstructed solely from their neighbours: $w_{i,j} = 0\, \forall j \notin \mathcal{N}(i)$
- Rows of **W** sum to one: $\sum_{j=1}^{N} w_{i,j} = 1$

Nice thing lies here

- Obtained $w_{i,j}$ verify invariance to rotations, scalings and translations of the associated point $\mathrm{y}(i)$ and its neighbours
- Hence, weights characterize intrinsic geometric properties of the considered neighbourhood of the manifold
- Hopefully, these geometric properties are also valid in a lower P-dimensional representation.

A little algorithm. . . (2)

To compute the $w_{i,j}$, $E(\mathbf{W})$ is minimized under two constraints

- Points are reconstructed solely from their neighbours: $w_{i,j} = 0\ \forall j \notin \mathcal{N}(i)$
- Rows of **W** sum to one: $\sum_{j=1}^{N} w_{i,j} = 1$

Nice thing lies here

- Obtained $w_{i,j}$ verify invariance to rotations, scalings and translations of the associated point $\mathbf{y}(i)$ and its neighbours
- Hence, weights characterize intrinsic geometric properties of the considered neighbourhood of the manifold
- Hopefully, these geometric properties are also valid in a lower P-dimensional representation.

A little algorithm. . . (2)

To compute the $w_{i,j}$, $E(\mathbf{W})$ is minimized under two constraints

- Points are reconstructed solely from their neighbours: $w_{i,j} = 0 \; \forall j \notin \mathcal{N}(i)$
- Rows of **W** sum to one: $\sum_{j=1}^{N} w_{i,j} = 1$

Nice thing lies here

- Obtained $w_{i,j}$ verify invariance to rotations, scalings and translations of the associated point $\mathbf{y}(i)$ and its neighbours
- Hence, weights characterize intrinsic geometric properties of the considered neighbourhood of the manifold
- Hopefully, these geometric properties are also valid in a lower P-dimensional representation.

A little algorithm. . . (2)

To compute the $w_{i,j}$, $E(\mathbf{W})$ is minimized under two constraints

- Points are reconstructed solely from their neighbours: $w_{i,j} = 0 \; \forall j \notin \mathcal{N}(i)$
- Rows of **W** sum to one: $\sum_{j=1}^{N} w_{i,j} = 1$

Nice thing lies here

- Obtained $w_{i,j}$ verify invariance to rotations, scalings and translations of the associated point $\mathbf{y}(i)$ and its neighbours
- Hence, weights characterize intrinsic geometric properties of the considered neighbourhood of the manifold
- Hopefully, these geometric properties are also valid in a lower P-dimensional representation.

A little algorithm. . . (2)

To compute the $w_{i,j}$, $E(\mathbf{W})$ is minimized under two constraints

- Points are reconstructed solely from their neighbours: $w_{i,j} = 0 \, \forall j \notin \mathcal{N}(i)$
- Rows of $\mathbf{W}$ sum to one: $\sum_{j=1}^{N} w_{i,j} = 1$

Nice thing lies here

- Obtained $w_{i,j}$ verify invariance to rotations, scalings and translations of the associated point $\mathbf{y}(i)$ and its neighbours
- Hence, weights characterize intrinsic geometric properties of the considered neighbourhood of the manifold
- Hopefully, these geometric properties are also valid in a lower P-dimensional representation.

A little algorithm. . . (3)

- Coordinates in this P-dimensional space are found by minimizing the embedding cost function

$$\Phi(\hat{\mathbf{X}}) = \sum_{i=1}^{N} \left\| \hat{\mathbf{x}}(i) - \sum_{j \in \mathcal{N}(i)} w_{i,j} \hat{\mathbf{x}}(j) \right\|^2 \tag{6}$$

- Error calculated in the embedding space, this time, and $w_{i,j}$ fixed
- Details of the calculation omitted. Use an EVD on a certain matrix $\mathbf{M} = (\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W})$ to minimize $\Phi(\hat{\mathbf{X}})$ and find the coordinates in the P-dimensional space

A little algorithm. . . (3)

- Coordinates in this P-dimensional space are found by minimizing the embedding cost function

$$\Phi(\hat{\mathbf{X}}) = \sum_{i=1}^{N} \left\| \hat{\mathbf{x}}(i) - \sum_{j \in \mathcal{N}(i)} w_{i,j} \hat{\mathbf{x}}(j) \right\|^2 \tag{6}$$

- Error calculated in the embedding space, this time, and $w_{i,j}$ fixed
- Details of the calculation omitted. Use an EVD on a certain matrix $M = (I - W)^T (I - W)$ to minimize $\Phi(\hat{\mathbf{X}})$ and find the coordinates in the P-dimensional space

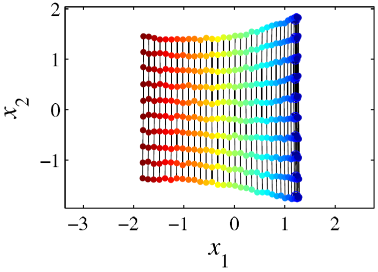
A little algorithm. . . (3)

- Coordinates in this P-dimensional space are found by minimizing the embedding cost function

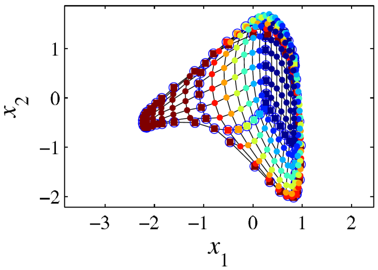
$$\Phi(\hat{\mathbf{X}}) = \sum_{i=1}^{N} \left\| \hat{\mathbf{x}}(i) - \sum_{j \in \mathcal{N}(i)} w_{i,j} \hat{\mathbf{x}}(j) \right\|^2 \tag{6}$$

- Error calculated in the embedding space, this time, and $w_{i,j}$ fixed
- Details of the calculation omitted. Use an EVD on a certain matrix $\mathbf{M} = (\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W})$ to minimize $\Phi(\hat{\mathbf{X}})$ and find the coordinates in the P-dimensional space

Looks rather nice



(m) Swiss roll: LLE



(n) Open box: LLE

Smooth, correct embedding, except for a crushed face on the open box

Advantages and some other things...

- Assumes that data linear locally, not globally
- $\Longrightarrow$ Manifold can be mapped to a plane using a conformal mapping
- Elegant for the mind and simple in the ideas
- Sticks to an eigensolver for the hardest part (and matrix often sparse, which makes things easier)

Ideas behind. . .

- Also going for a local approach to the problem of NLDR
- This time, minimization of neighbouring distances within the graph, with constraints (avoids the trivial case)
- Relies on the idea that the data set $\mathbf{Y} = \{\ldots, \mathbf{y}(i), \ldots, \mathbf{y}(j), \ldots\}_{1 \leq i,j \leq M}$ contains a sufficiently large number N of points on a smooth P-dimensional manifold
- If N large enough, manifold can be represented by a graph $G = (V_N, E)$
- Again, neighbourhood relationships determined using K-ary neighourhoods or ϵ-balls

Ideas behind. . .

- Also going for a local approach to the problem of NLDR
- This time, minimization of neighbouring distances within the graph, with constraints (avoids the trivial case)
- Relies on the idea that the data set $\mathbf{Y} = \{\ldots, \mathbf{y}(i), \ldots, \mathbf{y}(j), \ldots\}_{1 \leq i,j \leq M}$ contains a sufficiently large number N of points on a smooth P-dimensional manifold
- If N large enough, manifold can be represented by a graph $G = (V_N, E)$
- Again, neighbourhood relationships determined using K-ary neighourhoods or ϵ-balls

Ideas behind...

- Also going for a local approach to the problem of NLDR
- This time, minimization of neighbouring distances within the graph, with constraints (avoids the trivial case)
- Relies on the idea that the data set $\mathbf{Y} = \{\ldots, \mathbf{y}(i), \ldots, \mathbf{y}(j), \ldots\}_{1 \leq i,j \leq M}$ contains a sufficiently large number N of points on a smooth P-dimensional manifold
- If N large enough, manifold can be represented by a graph $G = (V_N, E)$
- Again, neighbourhood relationships determined using K-ary neighourhoods or ϵ-balls

Ideas behind...

- Also going for a local approach to the problem of NLDR
- This time, minimization of neighbouring distances within the graph, with constraints (avoids the trivial case)
- Relies on the idea that the data set $\mathbf{Y} = \{\ldots, \mathbf{y}(i), \ldots, \mathbf{y}(j), \ldots\}_{1 \leq i,j \leq M}$ contains a sufficiently large number N of points on a smooth P-dimensional manifold
- If N large enough, manifold can be represented by a graph $G = (V_N, E)$
- Again, neighbourhood relationships determined using K-ary neighourhoods or ϵ-balls

Ideas behind...

- Also going for a local approach to the problem of NLDR
- This time, minimization of neighbouring distances within the graph, with constraints (avoids the trivial case)
- Relies on the idea that the data set $\mathbf{Y} = \{\dots, \mathbf{y}(i), \dots, \mathbf{y}(j), \dots\}_{1 \leq i,j \leq M}$ contains a sufficiently large number N of points on a smooth P-dimensional manifold
- If N large enough, manifold can be represented by a graph $G = (V_N, E)$
- Again, neighbourhood relationships determined using K-ary neighourhoods or ϵ-balls

What's done in practice...

Some steps of LE

- Map $\mathbf{Y}$ to a set of low dimensional points $\mathbf{X} = \{\dots, \mathbf{x}(i), \dots, \mathbf{x}(j), \dots\}_{1 \leq i,j \leq N}$ keeping same neighbourhgood relationships, under the constraint of minimizing

$$E_{LE} = \frac{1}{2} \sum_{i,j=1}^{N} ||\mathbf{x}(i) - \mathbf{x}(j)||_2^2 w_{i,j} \tag{7}$$

  with $w_{i,j} = 0$ if $\mathbf{y}(i)$ and $\mathbf{y}(j)$ not neighbours, and $0 \leq w_{i,j}$ otherwise.

- Most often, $w_{i,j}$ follow a Gaussian kernel, or more simply, $w_{i,j} = 1$ if $\mathbf{y}(i)$ and $\mathbf{y}(j)$ are neighbours
- Thus, minimizing E_{LE} means that if $\mathbf{y}(i)$ and $\mathbf{y}(j)$ are close to each other, then $\mathbf{x}(i)$ and $\mathbf{x}(j)$ should be as well

What's done in practice. . .

Some steps of LE

- Map $\mathbf{Y}$ to a set of low dimensional points $\mathbf{X} = \{\ldots, \mathbf{x}(i), \ldots, \mathbf{x}(j), \ldots\}_{1 \leq i,j \leq N}$ keeping same neighbourhgood relationships, under the constraint of minimizing

$$E_{LE} = \frac{1}{2} \sum_{i,j=1}^{N} ||\mathbf{x}(i) - \mathbf{x}(j)||_2^2 w_{i,j} \tag{7}$$

  with $w_{i,j} = 0$ if $\mathbf{y}(i)$ and $\mathbf{y}(j)$ not neighbours, and $0 \leq w_{i,j}$ otherwise.

- Most often, $w_{i,j}$ follow a Gaussian kernel, or more simply, $w_{i,j} = 1$ if $\mathbf{y}(i)$ and $\mathbf{y}(j)$ are neighbours
- Thus, minimizing E_{LE} means that if $\mathbf{y}(i)$ and $\mathbf{y}(j)$ are close to each other, then $\mathbf{x}(i)$ and $\mathbf{x}(j)$ should be as well

What's done in practice...

Some steps of LE

- Map $\mathbf{Y}$ to a set of low dimensional points $\mathbf{X} = \{\ldots, \mathbf{x}(i), \ldots, \mathbf{x}(j), \ldots\}_{1 \leq i,j \leq N}$ keeping same neighbourhgood relationships, under the constraint of minimizing

$$E_{LE} = \frac{1}{2} \sum_{i,j=1}^{N} ||\mathbf{x}(i) - \mathbf{x}(j)||_2^2 w_{i,j} \tag{7}$$

  with $w_{i,j} = 0$ if $\mathbf{y}(i)$ and $\mathbf{y}(j)$ not neighbours, and $0 \leq w_{i,j}$ otherwise.

- Most often, $w_{i,j}$ follow a Gaussian kernel, or more simply, $w_{i,j} = 1$ if $\mathbf{y}(i)$ and $\mathbf{y}(j)$ are neighbours
- Thus, minimizing E_{LE} means that if $\mathbf{y}(i)$ and $\mathbf{y}(j)$ are close to each other, then $\mathbf{x}(i)$ and $\mathbf{x}(j)$ should be as well

What's done in practice. . . (2)

With some calculations, criterion reduces to

$$E_{LE} = tr(\mathbf{X}\mathbf{L}\mathbf{X}^T) \tag{8}$$

with $\mathbf{L} = \mathbf{W} - \mathbf{D}$ being the weighted Laplacian matrix of the graph G,

and $\mathbf{D}$ diagonal with $d_{i,j} = \sum_{j=1}^{N} w_{i,j}$

Problem ends up to an EVD of $\mathbf{L}$ and keep the P "lowest" eigenvectors

Algorithm in a big nutshell...

LE algorithm

1. Determine neighbourhoods (K-ary or ϵ-balls)
2. Build the graph (and determine adjacencies)
3. Build matrix $\mathbf{W}$ (using kernel or...)
4. Compute $\mathbf{D}$ matrix (diagonal, sums of weights rowwise)
5. Compute $\mathbf{L}$, Laplacian matrix of $\mathbf{W}$: $\mathbf{L} = \mathbf{W} - \mathbf{D}$
6. Normalize the Laplacian matrix
7. Compute its EVD and do some operations on eigenvectors to obtain the embedding

But again, life is cruel...

Parameters controlling the graph (K or ϵ) are very sensitive and require great care

Algorithm in a big nutshell. . .

LE algorithm

1. Determine neighbourhoods (K-ary or ϵ-balls)
2. Build the graph (and determine adjacencies)
3. Build matrix $\mathbf{W}$ (using kernel or. . .)
4. Compute $\mathbf{D}$ matrix (diagonal, sums of weights rowwise)
5. Compute $\mathbf{L}$, Laplacian matrix of $\mathbf{W}$: $\mathbf{L} = \mathbf{W} - \mathbf{D}$
6. Normalize the Laplacian matrix
7. Compute its EVD and do some operations on eigenvectors to obtain the embedding

But again, life is cruel. . .

Parameters controlling the graph (K or ϵ) are very sensitive and require great care

Algorithm in a big nutshell...

LE algorithm

1. Determine neighbourhoods (K-ary or ϵ-balls)
2. Build the graph (and determine adjacencies)
3. Build matrix $\mathbf{W}$ (using kernel or...)
4. Compute $\mathbf{D}$ matrix (diagonal, sums of weights rowwise)
5. Compute $\mathbf{L}$, Laplacian matrix of $\mathbf{W}$: $\mathbf{L} = \mathbf{W} - \mathbf{D}$
6. Normalize the Laplacian matrix
7. Compute its EVD and do some operations on eigenvectors to obtain the embedding

But again, life is cruel...

Parameters controlling the graph (K or ϵ) are very sensitive and require great care

Algorithm in a big nutshell. . .

LE algorithm

1. Determine neighbourhoods (K-ary or ϵ-balls)
2. Build the graph (and determine adjacencies)
3. Build matrix $\mathbf{W}$ (using kernel or. . .)
4. Compute $\mathbf{D}$ matrix (diagonal, sums of weights rowwise)
5. Compute $\mathbf{L}$, Laplacian matrix of $\mathbf{W}$: $\mathbf{L} = \mathbf{W} - \mathbf{D}$
6. Normalize the Laplacian matrix
7. Compute its EVD and do some operations on eigenvectors to obtain the embedding

But again, life is cruel. . .

Parameters controlling the graph (K or ϵ) are very sensitive and require great care

Algorithm in a big nutshell...

LE algorithm

1. Determine neighbourhoods (K-ary or ϵ-balls)
2. Build the graph (and determine adjacencies)
3. Build matrix $\mathbf{W}$ (using kernel or...)
4. Compute $\mathbf{D}$ matrix (diagonal, sums of weights rowwise)
5. Compute $\mathbf{L}$, Laplacian matrix of $\mathbf{W}$: $\mathbf{L} = \mathbf{W} - \mathbf{D}$
6. Normalize the Laplacian matrix
7. Compute its EVD and do some operations on eigenvectors to obtain the embedding

But again, life is cruel...

Parameters controlling the graph (K or ϵ) are very sensitive and require great care

Algorithm in a big nutshell...

LE algorithm

1. Determine neighbourhoods (K-ary or ϵ-balls)
2. Build the graph (and determine adjacencies)
3. Build matrix $\mathbf{W}$ (using kernel or...)
4. Compute $\mathbf{D}$ matrix (diagonal, sums of weights rowwise)
5. Compute $\mathbf{L}$, Laplacian matrix of $\mathbf{W}$: $\mathbf{L} = \mathbf{W} - \mathbf{D}$
6. Normalize the Laplacian matrix
7. Compute its EVD and do some operations on eigenvectors to obtain the embedding

But again, life is cruel...

Parameters controlling the graph (K or ϵ) are very sensitive and require great care

Algorithm in a big nutshell...

LE algorithm

1. Determine neighbourhoods (K-ary or ϵ-balls)
2. Build the graph (and determine adjacencies)
3. Build matrix $\mathbf{W}$ (using kernel or...)
4. Compute $\mathbf{D}$ matrix (diagonal, sums of weights rowwise)
5. Compute $\mathbf{L}$, Laplacian matrix of $\mathbf{W}$: $\mathbf{L} = \mathbf{W} - \mathbf{D}$
6. Normalize the Laplacian matrix
7. Compute its EVD and do some operations on eigenvectors to obtain the embedding

But again, life is cruel...

Parameters controlling the graph (K or ϵ) are very sensitive and require great care

Algorithm in a big nutshell...

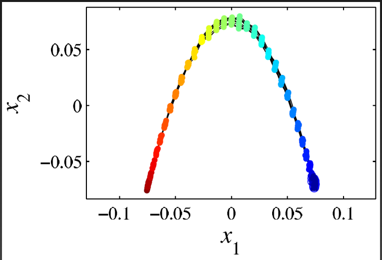
LE algorithm

1. Determine neighbourhoods (K-ary or ϵ-balls)
2. Build the graph (and determine adjacencies)
3. Build matrix **W** (using kernel or...)
4. Compute **D** matrix (diagonal, sums of weights rowwise)
5. Compute **L**, Laplacian matrix of **W**: $\mathbf{L} = \mathbf{W} - \mathbf{D}$
6. Normalize the Laplacian matrix
7. Compute its EVD and do some operations on eigenvectors to obtain the embedding

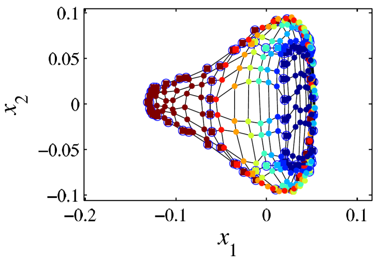
But again, life is cruel...

Parameters controlling the graph (K or ϵ) are very sensitive and require great care

Not-so-bad-but-not-so-good. . .



(o) Swiss roll: LE

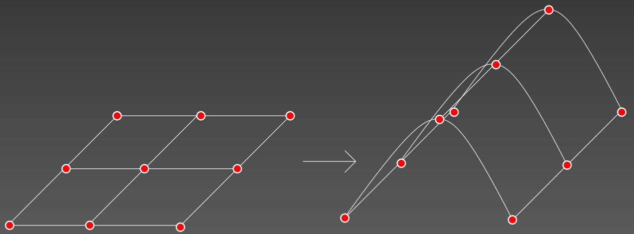


(p) Open box: LE

Swiss roll has third (spiral) dimension crushed and box is OK except for one crushed dimension again

Finally, for LE...

- LE has few parameters (once kernel for weights **W** is chosen): K or ϵ
- But parameters have a "dramatic" influence on results
- Apparently much nicer for clustering than dimensionality reduction
- Minimizing distances may lead to degenerate solutions (all in one point or such)



A small conclusion on these two methods...

- LLE has sexy and seducing ideas and concepts
- Not so hard on the calculation part
- Rather ok results (box crushed is "usual" unfortunately)

- LE is definately not meant for dimension reduction
- Seriously, the book says so!
- Used for clustering, more
- You may as well forget this one, I guess

And now, for the end of the show...

Antti takes on with Isotop

A small conclusion on these two methods...

- LLE has sexy and seducing ideas and concepts
- Not so hard on the calculation part
- Rather ok results (box crushed is "usual" unfortunately)

- LE is definately not meant for dimension reduction
- Seriously, the book says so!
- Used for clustering, more
- You may as well forget this one, I guess

And now, for the end of the show...

Antti takes on with Isotop

A small conclusion on these two methods...

- LLE has sexy and seducing ideas and concepts
- Not so hard on the calculation part
- Rather ok results (box crushed is "usual" unfortunately)

- LE is definately not meant for dimension reduction
- Seriously, the book says so!
- Used for clustering, more
- You may as well forget this one, I guess

And now, for the end of the show...

Antti takes on with Isotop