Introduction to Topological Data Analysis

Persistent Homology

Norm Matloff
University of California, Davis

Broad Overview

- Determine "what is connected to what" in dataset. Definition of *connected* depends on the application and the ingenuity of the analyst. **(Note this.)**
- Do this in each of a sequence of steps.
- Each step produces some kind of data summarizing connectivity. The data is collectively called a *filtration*.
- Use that output data as features, e.g. to do classification.

Image Classification Example

Image Classification Example

- The famous MNIST data, hand-drawn digits. Determine what digit it is, by analyzing the pixels (28×28).
- Not just greyscale, but mainly black-and-white. Here I'll look only a pixels > 192 level.
- For simplicity, I'll first use a somewhat nonstandard (and new-ish) TDA method.
  - May or may not be better than other methods.
  - But is simple, easy to explain and draw.
  - **Just an example**.

Crucial need for Dimension Reduction

Crucial need for Dimension Reduction

- In MNIST case, we are predicting digit from $28^2 = 784$ features.
- 784 way too large: (a) Overfitting. (b) Horrendous computation needs.
- So, we need to convert the existing 784 features to a smaller number (*dimension reduction*). But how?

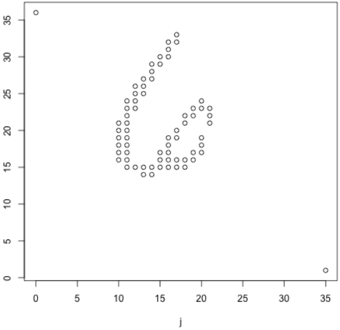
Dimension Reduction Methods for Images

Dimension Reduction Methods for Images

- Principal Components Analysis (PCA)
  - A traditional approach. Project the data from R^{784} to, say, R^{50}, using eigenanalysis.
  - Plug into logit, maybe with polynomial terms (my **polyreg** package).
- Convolutional Neural Networks (CNNs)
  - Currently most fashionable.
  - Not new! The "C" part of CNN is just **traditional image smoothing**, breaking the image into small tiles, and then e.g. finding the median pixel intensity in each tile. E.g. in MNIST, take 4×4 tiles, so now have $7^2 = 49$ predictors.
- Geometric methods:
  - Runs statistics (counts of how many consecutive vertical or horizontal pixels are black, etc.).
  - TDA.

A '6'

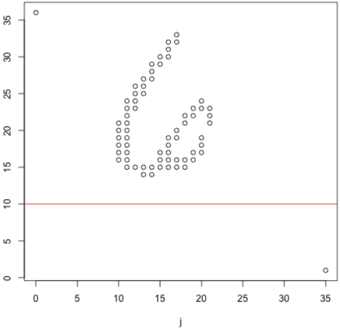
A '6'



Filtration plan:

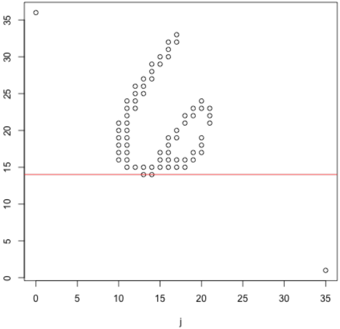
- Draw a series of horizontal lines.
- See how many components are formed in the figure by a line.

A '6'



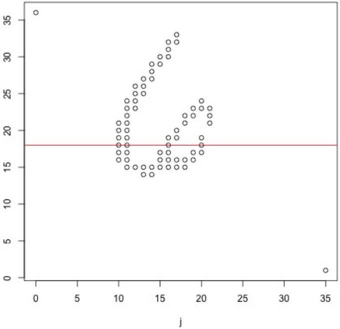
0 components

A '6'



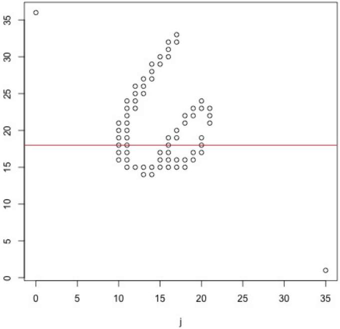
1 component (2 adjacent pixels)

A '6'



3 components (2 adj. pixels, then 1 and 1)

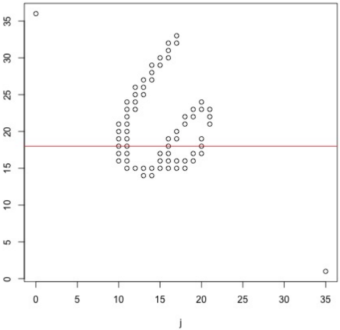
A '6'



3 components (2 adj. pixels, then 1 and 1)

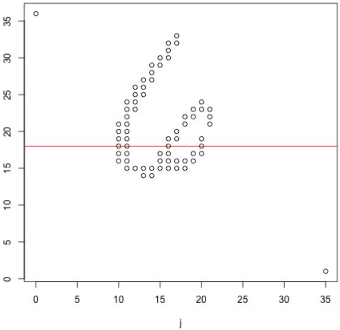
Birth, Death Times

Birth, Death Times



Then as the red line is moved upward, will mostly have 3 components for a while, then 1.

Birth, Death Times

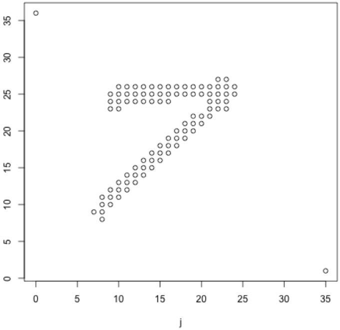


Then as the red line is moved upward, will mostly have 3 components for a while, then 1.

We talk about *birth* and *death* times. E.g. the first 3-component line is "born" at line 17 and "dies" at line 25.

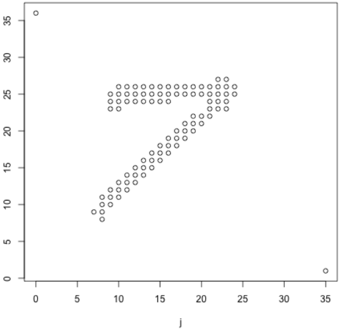
A '7'

A '7'



A 1-component line will be born early on, then persist for a long time.

A '7'



A 1-component line will be born early on, then persist for a long time.
Then we may get a 2-component birth, not long-lived.

'6' vs. '7'

'6' vs. '7'

digit	pattern
'6'	3 comps., then 1
'7'	1 comp., then 2

'6' vs. '7'

digit	pattern
'6'	3 comps., then 1
'7'	1 comp., then 2

- So, easy to distinguish '6' and '7' via BD data, right?

'6' vs. '7'

digit	pattern
'6'	3 comps., then 1
'7'	1 comp., then 2

- So, easy to distinguish '6' and '7' via BD data, right?
- But what if the top bar of a '7' is angled slightly up, not down?

'6' vs. '7'

digit	pattern
'6'	3 comps., then 1
'7'	1 comp., then 2

- So, easy to distinguish '6' and '7' via BD data, right?
- But what if the top bar of a '7' is angled slightly up, not down? Then only have a 1-comp.

'6' vs. '7'

digit	pattern
'6'	3 comps., then 1
'7'	1 comp., then 2

- So, easy to distinguish '6' and '7' via BD data, right?
- But what if the top bar of a '7' is angled slightly up, not down? Then only have a 1-comp.

A Second Opinion

A Second Opinion

Solution: "Get a second opinion": Collect vertical-bar BD data.

digit	pattern
'6'	mainly 3 comps.
'7'	mainly 2 comps.

A Second Opinion

Solution: “Get a second opinion”: Collect vertical-bar BD data.

digit	pattern
'6'	mainly 3 comps.
'7'	mainly 2 comps.

So, our new features could be the two sets of BD data, horizontal and vertical sweeps.

Not Out of the Woods Yet

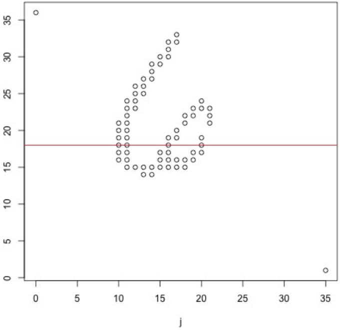
Not Out of the Woods Yet

Not so simple. For instance:

- **Anomalous BDs:** Sometimes have fainter pixels than our 192 threshold. E.g. line 20 in the '6' had a gap. Causes an incorrect birth/death.
- **Vectorization:** Different images for the same digit have different numbers of BD data. But ML methods require the feature vector to have a constant number of features from one data point to another (in this case one image to another).
- **Orientation:**The above filtration scheme largely assumed:
  - Mainly black-and-white image, not even greyscale (e.g. Fashion MNIST).
  - Image has a notion of left-right, up-down.

Possible Solutions: Anomalous BDs

Possible Solutions: Anomalous BDs



- Ignore row 20 in the BD calculation.
- Ignore any row/column that would create a short-lived component (D - B = 1 or 2, say).
- But what if they are real?
- Maybe do BD at each of several pixel intensity thresholds,

Possible Solutions: Vectorization

Possible Solutions: Vectorization

- Say have 35-row images. The possible (B,D) grid is $(i,j) : 1 \leq i < j \leq 35)$. For each image, calculate the count of (B,D) pairs at each grid point, as the red horizontal line moves up. Do the same for the red vertical lines. That data, placed in a vector, is now the feature vector for this image.
- For a large, detailed image, the above method may need voluminous computation and/or lead to overfitting. Some analysts devise their own *ad hoc* method. E.g. Garside (2019) compute a vector consisting of the number of pixels, average lifetime, area under the persistence function, and four measures based on polygons drawn in the graph of persistence.

Possible Solutions: Orientation

Possible Solutions: Orientation

Lots of filtration methods don't assume the image has a left and right, up and down.
E.g. "topographic" method (described next).

“Topographic” Filtration

"Topographic" Filtration

- Here the thresholding on pixel intensity *is* the filtration, rather than an add-on as above.

"Topographic" Filtration

- Here the thresholding on pixel intensity *is* the filtration, rather than an add-on as above.
- Imagine a 3-D representation of image. X, Y dims. are image row, column, then Z is the pixel intensity. Looks like mountain peaks above the (X,Y) plane.
- Instead of a red line, we now have a red plane, above and parallel to the (X,Y).
- Initially, all nonzero pixels are above the red plane. But as it moves higher, the pixels with lower intensities begin to drop out, thus creating BD data.
- No implied notion of left-right, up-down.
- Again, 3 sets of BD data for RGB.

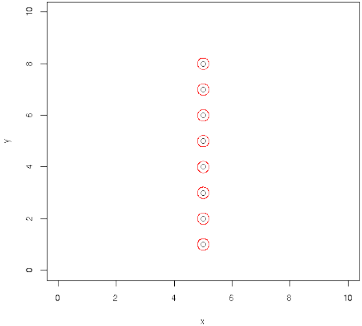
(Vietoris-)Rips Filtrations

(Vietoris-)Rips Filtrations

- Draw a red circle around each data point, same radius for all.
- The filtration consists of drawing an increasing sequence of radii.
- Points in overlapping circles are considered to be in the same component.

An 'I'

An 'I'

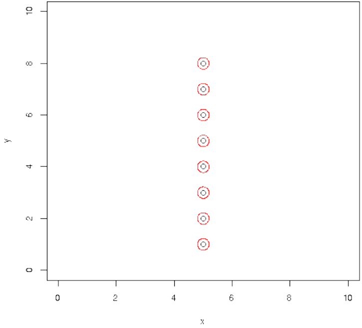


- radius 0.2
- 8 components

Norm Matloff
University of California, Davis

An 'I'

An 'I'

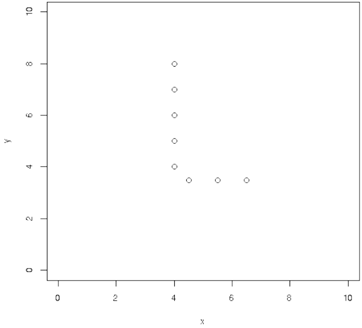


- radius 0.6
- 1 component
- the 8 components died at 0.5, the 1 component was born at 0.5

Norm Matloff
University of California, Davis

An 'L'

An 'L'

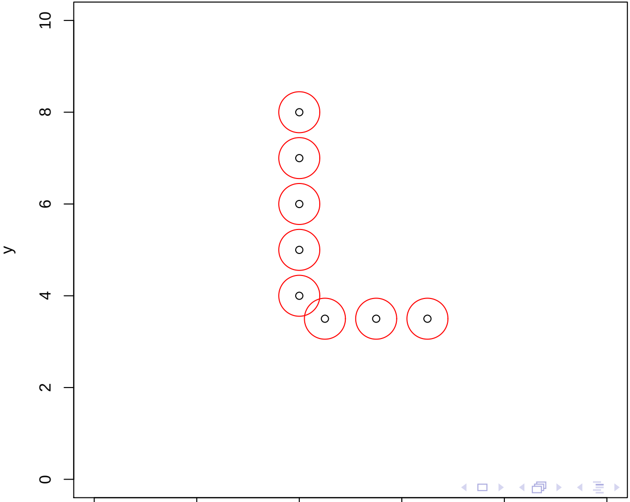


- I took the 'I' and just bent it; linear distance between points still 1.0
- but now there will be a birth at $0.5(0.5\sqrt{2}) = 0.35$, not 0.5
- originally 8 components, then 7, then 1

An 'L'

An 'L'

Radius 0.4:



Rips Senses Angles!

Rips Senses Angles!

The point:

> *Rips filtration does more than topology; it does geometry. (Math: curvature)*

Vectorization

Vectorization