Clustering

Steve Oudot

(steve.oudot@inria.fr)
Cluster Analysis

**Input:** a finite set of observations:  
- point cloud with coordinates
- distance / (dis-)similarity matrix

**Task:**
partition the data points into a collection of *relevant* subsets called clusters
A Wealth of Approaches

Variational
- $k$-means / $k$-medoid
- EM
- CLARA

Spectral
- Normalized Cut
- Multiway Cut

Hierarchical divisive/agglomerative
- single-linkage
- BIRCH

Density thresholding
- DBSCAN
- OPTICS

Mode seeking
- Mean/Medoid/Quick Shift
- graph-based hill climbing

Valley seeking
- [JBD’79]
- NDDs [ZZZL’07]
A Wealth of Approaches

Variational
- $k$-means / $k$-medoid
- EM
- CLARA

Spectral
- Normalized Cut
- Multiway Cut

Hierarchical divisive/agglomerative
- single-linkage
- BIRCH

Density thresholding
- DBSCAN
- OPTICS

Mode seeking
- Mean/Medoid/Quick Shift
- graph-based hill climbing

Valley seeking
- [JBD’79]
- NDDs [ZZZL’07]
Mode-Seeking Paradigm

- Assume the data points are sampled from some unknown probability distribution
- Partition the data according to the basins of attraction of the peaks of the density
Mode-Seeking Paradigm

- Assume the data points are sampled from some unknown probability distribution
- Partition the data according to the basins of attraction of the peaks of the density
Mode-Seeking Paradigm

- Assume the data points are sampled from some unknown probability distribution
- Partition the data according to the basins of attraction of the peaks of the density
Mode-Seeking Paradigm

- Assume the data points are sampled from some unknown probability distribution
- Partition the data according to the basins of attraction of the peaks of the density
Mode-Seeking Paradigm

- Assume the data points are sampled from some unknown probability distribution
- Partition the data according to the basins of attraction of the peaks of the density
[Koontz, Narendra, Fukunaga’76] in a Nutshell
typically, one uses a Gaussian kernel estimator in practice to estimate density at the data points.

[Koontz, Narendra, Fukunaga’76] in a Nutshell
[Koontz, Narendra, Fukunaga’76] in a Nutshell

estimate density at the data points
[Koontz, Narendra, Fukunaga’76] in a Nutshell

1. Build neighborhood graph
2. Estimate density at the data points
3. Approximate gradient by a graph edge at each data point

(Koontz, Narendra, Fukunaga’76) in a Nutshell
Why things are likely to go ill

- Noisy estimator

The main reason why we got a wrong result here is that our estimator is very noisy, with many local peaks in the plane. Generally speaking, differential quantities like peaks and gradients are very unstable under $C^0$ perturbations of the function, which is what happens when a density estimator is used.

The result obtained depends on your choice of estimator, neighborhood graph and gradient approximation strategy.
Why things are likely to go ill

- Noisy estimator
- Neighborhood graph
Why things are likely to go ill

- Noisy estimator
- Neighborhood graph

Solutions:

1. **Be proactive**: act on approximate gradient flow ([Mean-Shift [CM’02]])
   - use kernel density estimator, with smoothing window parameter
   - work in ambient space to circumvent neighborhood graph issue
Why things are likely to go ill

• Noisy estimator

• Neighborhood graph

Solutions:

1. **Be proactive**: act on approximate gradient flow (**Mean-Shift** [CM’02])
   → use kernel density estimator, with smoothing window parameter
   → work in ambient space to circumvent neighborhood graph issue

2. **Be reactive**: merge clusters after clustering (**ToMATo** [CGOS’13])
   → use topological persistence to guide a single-pass merging step
   → work in neighborhood graph to minimize prior knowledge
1. Mean-Shift
Kernel density estimators

**Principle:** take a mixture of copies of an ‘elementary’ density (kernel), anchored at each observation.
Kernel density estimators

**Principle:** take a mixture of copies of an ‘elementary’ density (kernel), anchored at each observation

(image source: http://www.wikiwand.com/en/Multivariate_kernel_density_estimation)
Kernel density estimators

**Input:** \( P = \{p_1, \cdots, p_n\} \subset \mathbb{R}^d \) (data points), \( x \in \mathbb{R}^d \) (query point)

**General formula:** (convolution)

\[
\hat{f}_{KH}(x) := \frac{1}{n} \sum_{i=1}^{n} K_H(x - p_i), \quad \text{where} \quad K_H(u) := (\det H)^{-1/2} K(H^{-1/2}u)
\]

- \( H \): inner-product (positive-definite) \( d \times d \) matrix (adds scaling / anisotropy)
- \( K \): \( \mathbb{R}^d \rightarrow \mathbb{R}^+ \): \( d \)-variate kernel:

  \[
  \int_{\mathbb{R}^d} K(u) \, du = 1 \quad \text{(normalized)} \quad \int_{\mathbb{R}^d} u K(u) \, du = 0 \quad \text{(centered at origin)}
  \]

  \[
  \lim_{\|u\| \rightarrow \infty} K(u) = 0 \quad \text{(vanishes at infinity)} \quad \int_{\mathbb{R}^d} uu^T K(u) \, du = c_K I_d \quad \text{(isotropic)}
  \]
Kernel density estimators

**Specialization 1:** take $H = \sigma^2 I_d$ (isotropic kernel)

bandwidth / window
Kernel density estimators

Specialization 1: take \( H = \sigma^2 I_d \) (isotropic kernel)

Kernel-based estimators have been designed to adapt naturally to the shape of the support of the density, as the corresponding "tessellation" (so to speak) is defined from the data.

Specialization 2: take \( K(u) \propto k(\|u\|_2^2) \) for some \( k : \mathbb{R}^+ \rightarrow \mathbb{R}^+ \) (radially-symmetric kernel)

normalizing factor: \( c_{k,d} := \left( \int_{\mathbb{R}^d} k(\|u\|_2^2) \, du \right)^{-1} \)
Kernel density estimators

Specialization 1: take \( H = \sigma^2 I_d \) (isotropic kernel)

Specialization 2: take \( K(u) \propto k(\|u\|_2^2) \) for some \( k : \mathbb{R}^+ \rightarrow \mathbb{R}^+ \)

(normalizing factor: \( c_{k,d} := \left( \int_{\mathbb{R}^d} k(\|u\|_2^2) \, du \right)^{-1} \))

\[
\hat{f}_{\sigma,k}(x) := \frac{c_{k,d}}{n \sigma^d} \sum_{i=1}^{n} k\left(\frac{\|x - p_i\|_2^2}{\sigma^2}\right)
\]
Common kernels

Flat / Uniform: $k_U(t) := \begin{cases} 
1 & \text{if } t \leq 1 \\
0 & \text{if } t > 1 
\end{cases}$

$\Rightarrow c_{k,d} = \frac{1}{\text{Vol} B_d(0,1)}$

$= \frac{\Gamma(d/2 + 1)}{\pi^{d/2}}$
Common kernels

**Flat / Uniform:** 
\[
k_{U}(t) := \begin{cases} 
1 & \text{if } t \leq 1 \\
0 & \text{if } t > 1
\end{cases}
\]
\[\Rightarrow c_{k,d} = \frac{1}{\text{Vol } B_{d}(0, 1)} = \frac{\Gamma(d/2 + 1)}{\pi^{d/2}}\]

**Epanechnikov:** 
\[
k_{E}(t) := \begin{cases} 
1 - t & \text{if } t \leq 1 \\
0 & \text{if } t > 1
\end{cases}
\]
\[\Rightarrow c_{k,d} = \frac{d + 2}{2 \cdot \text{Vol } B_{d}(0, 1)}\]
Common kernels

**Gaussian:** \( k_N(t) := \exp(-t/2) \)

\[ c_{k,d} = (2\pi)^{-d/2} \]

**Epanechnikov:** \( k_E(t) := \begin{cases} 1 - t & \text{if } t \leq 1 \\ 0 & \text{if } t > 1 \end{cases} \)

\[ c_{k,d} = \frac{d + 2}{2 \text{Vol } B_d(0, 1)} \]
Common kernels

Old faithful geyser dataset (available in R):
- 1st coordinate: waiting time (sec.) between eruptions
- 2nd coordinate (unused): eruptions duration (sec.)
Influence of the bandwidth

- small $\sigma$ (*undersmoothing*): small bias (sensitivity), large variance (instability)
- large $\sigma$ (*oversmoothing*): large bias (insensitivity), small variance (stability)

Old geyser dataset
Differentiation

\[
\hat{f}_{\sigma,k}(x) := \frac{c_{k,d}}{n \sigma^d} \sum_{i=1}^{n} k\left(\frac{\|x - p_i\|^2}{\sigma^2}\right)
\]

\[
\hat{\nabla} f(x) := \nabla \hat{f}_{\sigma,k}(x) = \frac{2c_{k,d}}{n \sigma^{d+2}} \sum_{i=1}^{n} (x - p_i) k'\left(\frac{\|x - p_i\|^2}{\sigma^2}\right)
\]
Differentiation

\[ \hat{f}_{\sigma,k}(x) := \frac{c_{k,d}}{n \sigma^d} \sum_{i=1}^{n} k \left( \frac{\|x - p_i\|^2}{\sigma^2} \right) \]

\[ \hat{\nabla} f(x) := \nabla \hat{f}_{\sigma,k}(x) = \frac{2 \, c_{k,d}}{n \sigma^{d+2}} \sum_{i=1}^{n} (x - p_i) \, k' \left( \frac{\|x - p_i\|^2}{\sigma^2} \right) \]

Letting \( g := -k' \) (assumed to be \( \geq 0 \)):

\[ \nabla \hat{f}_{\sigma,k}(x) = \frac{2 \, c_{k,d}}{n \sigma^{d+2}} \left( \sum_{i=1}^{n} g \left( \frac{\|x - p_i\|^2}{\sigma^2} \right) \right) \left( \frac{\sum_{i=1}^{n} p_i \, g \left( \frac{\|x - p_i\|^2}{\sigma^2} \right)}{\sum_{i=1}^{n} g \left( \frac{\|x - p_i\|^2}{\sigma^2} \right)} - x \right) \]
Differentiation

\[ \hat{f}_{\sigma,k}(x) := \frac{c_{k,d}}{n \sigma^d} \sum_{i=1}^{n} k\left(\frac{\|x - p_i\|^2}{\sigma^2}\right) \]

\[ \nabla \hat{f}(x) := \nabla \hat{f}_{\sigma,k}(x) = \frac{2 c_{k,d}}{n \sigma^{d+2}} \sum_{i=1}^{n} (x - p_i) k'\left(\frac{\|x - p_i\|^2}{\sigma^2}\right) \]

Letting \( g := -k' \) (assumed to be \( \geq 0 \)):

\[ \nabla \hat{f}_{\sigma,k}(x) = \frac{2 c_{k,d}}{n \sigma^{d+2}} \left( \sum_{i=1}^{n} g\left(\frac{\|x - p_i\|^2}{\sigma^2}\right) \right) \left( \frac{\sum_{i=1}^{n} p_i g\left(\frac{\|x-p_i\|^2}{\sigma^2}\right)}{\sum_{i=1}^{n} g\left(\frac{\|x-p_i\|^2}{\sigma^2}\right)} - x \right) \]

(un-normalized) kernel density estimator with profile \( g \)

barycenter w.r.t. \( g \)

mean-shift \( m_{\sigma,g}(x) \)
Differentiation

\[ \hat{f}_{\sigma,k}(x) := \frac{c_{k,d}}{n \sigma^d} \sum_{i=1}^{n} k\left( \frac{\|x - p_i\|_2^2}{\sigma^2} \right) \]

\[ \hat{\nabla} f(x) := \hat{\nabla} \hat{f}_{\sigma,k}(x) = \frac{2 c_{k,d}}{n \sigma^{d+2}} \sum_{i=1}^{n} (x - p_i) k'\left( \frac{\|x - p_i\|_2^2}{\sigma^2} \right) \]

Letting \( g := -k' \) (assumed to be \( \geq 0 \)):

\[ \nabla f_{\sigma,k}(x) = \frac{2 c_{k,d}}{n \sigma^{d+2}} \left( \sum_{i=1}^{n} g\left( \frac{\|x - p_i\|_2^2}{\sigma^2} \right) \right) \left( \frac{\sum_{i=1}^{n} p_i g\left( \frac{\|x - p_i\|_2^2}{\sigma^2} \right)}{\sum_{i=1}^{n} g\left( \frac{\|x - p_i\|_2^2}{\sigma^2} \right)} - x \right) \]

(un-normalized) kernel density estimator with profile \( g \)

barycenter w.r.t. \( g \)

mean-shift \( m_{\sigma,g}(x) \)

\( \Rightarrow \) gradient of density is collinear with mean-shift and oriented in the same direction
Mean-Shift

hill-climbing

**Input:** $P = \{p_1, \cdots, p_n\} \subset \mathbb{R}^d$ (data points), $x \in \mathbb{R}^d$ (query point to be labeled)

**Parameters:** $k : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ (profile), $\sigma > 0$ (bandwidth)

$x_0 := x$

Repeat:

$x_{j+1} := x_j + m_\sigma, g(x_j)$

until convergence

**Output:** the label associated with the convergence point
Mean-Shift

- Apply Mean-Shift hill-climbing to each input point $p_i \in P$
- Epanechnikov kernel $\Rightarrow$ convergence in finite time
  $\rightarrow$ may converge outside the set of critical points of the estimator
  $\rightarrow$ use variant to guarantee convergence to maximum [Huang et al. 2017]
Mean-Shift

- Apply Mean-Shift hill-climbing to each input point $p_i \in P$

- Epanechnikov kernel $\Rightarrow$ convergence in finite time
  - $\rightarrow$ may converge outside the set of critical points of the estimator
  - $\rightarrow$ use variant to guarantee convergence to maximum [Huang et al. 2017]

- Gaussian kernel $\Rightarrow$ convergence at the limit (infinite time)
  - $\rightarrow$ stopping criterion (convergence radius)
  - $\rightarrow$ identification of modes (mode radius)
  - $\rightarrow$ speed-up: hill-climbing gathers neighboring points (gathering radius)

$\Rightarrow$ heuristic: make these radii proportional to the estimator’s bandwidth $\sigma$
Examples [Comaniciu, Meer 2002]
Examples [Comaniciu, Meer 2002]
2. ToMATATo
[Koontz, Narendra, Fukunaga’76] in a Nutshell
[Koontz, Narendra, Fukunaga’76] in a Nutshell

estimate density at the data points
[Koontz, Narendra, Fukunaga’76] in a Nutshell

Typically, one builds a Rips or k-NN graph in practice, since these only require to use distance computations.

Typically, one uses a Gaussian kernel estimator in practice to estimate density at the data points.

Build neighborhood graph.

[Koontz, Narendra, Fukunaga’76]
typically, one connects each vertex to its graph neighbor with highest density value. This neighbor is called the parent of the current vertex. If no neighbor is higher than the current vertex, then the latter is declared a peak. Note that [KNF'76] normalizes the difference in height by the edge length.

The set of pseudo-gradient edges forms a spanning forest of the graph, where each tree represents a cluster and its root is a (estimated) density peak within the graph and acts as cluster center.

Typically, one builds a Rips or k-NN graph in practice, since these only require to use distance computations.

Typically, one uses a Gaussian kernel estimator in practice.

Estimate density at the data points.

Approximate gradient by a graph edge at each data point.
Pseudo-code:

**Input:** neighborhood graph $G$ with $n$ vertices, $n$-dimensional vector $\hat{f}$ (density estimator)

Sort the vertex indices $\{1, 2, \cdots, n\}$ so that $\hat{f}(1) \geq \hat{f}(2) \geq \cdots \geq \hat{f}(n)$; Initialize a union-find data structure (disjoint-set forest) $U$ and two vectors $g, r$ of size $n$;

for $i = 1$ to $n$ do
  Let $\mathcal{N}$ be the set of neighbors of $i$ in $G$ that have indices lower than $i$;
  if $\mathcal{N} = \emptyset$ // vertex $i$ is a peak of $\hat{f}$ within $G$
    Create a new entry $e$ in $U$ and attach vertex $i$ to it;
    $r(e) \leftarrow i$ // $r(e)$ stores the root vertex associated with the entry $e$
  else // vertex $i$ is not a peak of $\hat{f}$ within $G$
    $g(i) \leftarrow \text{argmax}_{j \in \mathcal{N}} \hat{f}(j)$ // $g(i)$ stores the approximate gradient at vertex $i$
    $e_i \leftarrow U.\text{find}(g(i))$;
    Attach vertex $i$ to the entry $e_i$;

**Output:** the collection of entries $e$ in $U$
Enter Topological Persistence...
Topological Persistence (in a nutshell)

$X$ topological space

$f : X \rightarrow \mathbb{R}$

persistence

$\text{D}g\ f$

signature: *persistence diagram*

encodes the topological structure of the pair $(X, f)$
Topological Persistence (in a nutshell)

Inside the black box:
- Nested family \((\text{filtration})\) of sublevel-sets \(f^{-1}((\infty, t])\) for \(t\) ranging from \(-\infty\) to \(+\infty\)
- Track the evolution of the topology throughout the family
Topological Persistence (in a nutshell)

Inside the black box:
- Nested family (filtration) of sublevel-sets $f^{-1}((-\infty, t])$ for $t$ ranging from $-\infty$ to $+\infty$
- Track the evolution of the topology throughout the family
Topological Persistence (in a nutshell)

Inside the black box:
- Nested family (filtration) of sublevel-sets $f^{-1}((-\infty, t])$ for $t$ ranging from $-\infty$ to $+\infty$
- Track the evolution of the topology throughout the family
Topological Persistence (in a nutshell)

Inside the black box:

- Nested family (*filtration*) of sublevel-sets $f^{-1}((−\infty, t])$ for $t$ ranging from $−\infty$ to $+\infty$
- Track the evolution of the topology throughout the family
Inside the black box:

- Nested family (filtration) of sublevel-sets $f^{-1}((-\infty, t])$ for $t$ ranging from $-\infty$ to $+\infty$
- Track the evolution of the topology throughout the family
Topological Persistence (in a nutshell)

Inside the black box:

- Nested family (filtration) of sublevel-sets $f^{-1}((-\infty, t])$ for $t$ ranging from $-\infty$ to $+\infty$
- Track the evolution of the topology throughout the family
Topological Persistence (in a nutshell)

Inside the black box:

- Nested family (filtration) of sublevel-sets $f^{-1}((\mathbb{R}, t])$ for $t$ ranging from $-\infty$ to $+\infty$
- Track the evolution of the topology throughout the family
Topological Persistence (in a nutshell)

Inside the black box:
- Nested family (*filtration*) of sublevel-sets $f^{-1}((−∞, t])$ for $t$ ranging from $−∞$ to $+∞$
- Track the evolution of the topology throughout the family
Topological Persistence (in a nutshell)

Inside the black box:

- Nested family (filtration) of sublevel-sets $f^{-1}((−\infty, t])$ for $t$ ranging from $−\infty$ to $+\infty$
- Track the evolution of the topology throughout the family
- Finite set of intervals (barcode) encodes births/deaths of topological features
Topological Persistence (in a nutshell)

Inside the black box:
- Nested family (filtration) of sublevel-sets $f^{-1}((-\infty, t])$ for $t$ ranging from $-\infty$ to $+\infty$
- Track the evolution of the topology throughout the family
- Finite set of intervals (barcode) encodes births/deaths of topological features

Alternate representation as a multiset of points in the plane (diagram).
Topological Persistence (in a nutshell)

Algorithm:

- input: graph $G = (V, E)$ + map $f : V \sqcup E \to \mathbb{R}$
- procedure: scan graph by increasing $f$-values, update CCs by union-find
Topological Persistence (in a nutshell)

Inside the black box:
- Nested family (filtration) of sublevel-sets $f^{-1}((-\infty, t])$ for $t$ ranging from $-\infty$ to $+\infty$
- Track the evolution of the topology throughout the family
- Finite set of intervals (barcode) encodes births/deaths of topological features

What if $f$ is slightly perturbed?

- Alternate representation as a multiset of points in the plane (diagram).
For any tame functions $f, g : \mathbb{X} \to \mathbb{R}$, $d^\infty_B(Dg f, Dg g) \leq \|f - g\|_\infty$.

partial matching $M : Dg f \leftrightarrow Dg g$

cost of a matched pair $(p, q) \in M$: $\|p - q\|_\infty$

cost of an unmatched point $s \in Dg f \sqcup Dg g$: $\|s - \bar{s}\|_\infty$

cost of a matching:

$$\max \left\{ \sup_{(p, q) \text{ matched}} \|p - q\|_\infty, \sup_{s \text{ unmatched}} \|s - \bar{s}\|_\infty \right\}$$

bottleneck distance:

$$d^\infty_B(Dg f, Dg g) = \inf_{M : Dg f \leftrightarrow Dg g} \text{cost}(M)$$
Example: Distance Function

\[ f_P : \mathbb{R}^2 \rightarrow \mathbb{R} \]
\[ x \mapsto \min_{p \in P} \|x - p\|_2 \]
Example: Distance Function

\[ f_P: \mathbb{R}^2 \rightarrow \mathbb{R} \]
\[ x \mapsto \min_{p \in P} \|x - p\|_2 \]
Example: Distance Function

\[ f_P : \mathbb{R}^2 \rightarrow \mathbb{R} \]
\[ x \mapsto \min_{p \in P} \| x - p \|_2 \]
Example: Distance Function

\[ f_P : \mathbb{R}^2 \to \mathbb{R} \]
\[ x \mapsto \min_{p \in P} \| x - p \|_2 \]
Example: Distance Function

\[ f_P : \mathbb{R}^2 \rightarrow \mathbb{R} \]
\[ x \mapsto \min_{p \in P} \| x - p \|_2 \]
Example: Distance Function

\[ f_P : \mathbb{R}^2 \to \mathbb{R} \]
\[ x \mapsto \min_{p \in P} \| x - p \|_2 \]
Example: Distance Function

\[ f_P : \mathbb{R}^2 \rightarrow \mathbb{R} \]
\[ x \mapsto \min_{p \in P} \| x - p \|_2 \]
Example: Distance Function

\[ f_P : \mathbb{R}^2 \rightarrow \mathbb{R} \]
\[ x \mapsto \min_{p \in P} \|x - p\|_2 \]
Example: Distance Function

\[ f_P : \mathbb{R}^2 \rightarrow \mathbb{R} \]
\[ x \mapsto \min_{p \in P} \|x - p\|_2 \]

\( \text{barcode} \rightarrow \text{merge tree} \)
Example: Distance Function

\[ f_P : \mathbb{R}^2 \to \mathbb{R} \]
\[ x \mapsto \min_{p \in P} \| x - p \|_2 \]
Back to Mode Seeking

(use density estimator instead of distance function)
Persistence for Mode Seeking

Given a probability density $f$:

- Nested family (filtration) of **superlevel-sets** $f^{-1}(\lbrack t, +\infty \rbrack)$ for $t$ from $+\infty$ to $-\infty$.
- Track evolution of topology throughout the family.
Persistence for Mode Seeking

Given a probability density $f$:

- Nested family (filtration) of superlevel-sets $f^{-1}([t, +\infty))$ for $t$ from $+\infty$ to $-\infty$.
- Track evolution of topology throughout the family.
Persistance for Mode Seeking

Given a probability density \( f \):
- Nested family (filtration) of **superlevel-sets** \( f^{-1}([t, +\infty)) \) for \( t \) from \(+\infty\) to \(-\infty\).
- Track evolution of topology throughout the family.
Persistence for Mode Seeking

Given a probability density $f$:

- Nested family (filtration) of superlevel-sets $f^{-1}([t, +\infty))$ for $t$ from $+\infty$ to $-\infty$.
- Track evolution of topology throughout the family.
Persistence for Mode Seeking

Given a probability density $f$:

- Nested family (filtration) of superlevel-sets $f^{-1}([t, +\infty))$ for $t$ from $+\infty$ to $-\infty$.
- Track evolution of topology throughout the family.
Persistence for Mode Seeking

Given a probability density $f$:

- Nested family (filtration) of superlevel-sets $f^{-1}([t, +\infty))$ for $t$ from $+\infty$ to $-\infty$.
- Track evolution of topology throughout the family.
Persistence for Mode Seeking

Given a probability density \( f \):
- Nested family (filtration) of superlevel-sets \( f^{-1}([t, +\infty)) \) for \( t \) from \( +\infty \) to \( -\infty \).
- Track evolution of topology throughout the family.
Persistence for Mode Seeking

Given a probability density $f$:
- Nested family (filtration) of superlevel-sets $f^{-1}([t, +\infty))$ for $t$ from $+\infty$ to $-\infty$.
- Track evolution of topology throughout the family.
- Finite set of intervals (barcode) encodes births/deaths of topological features.
Persistence for Mode Seeking

Given a probability density $f$:

- Nested family (filtration) of superlevel-sets $f^{-1}([t, +\infty))$ for $t$ from $+\infty$ to $-\infty$.
- Track evolution of topology throughout the family.
- Finite set of intervals (barcode) encodes births/deaths of topological features.
Persistence for Mode Seeking

Given an estimator $\hat{f}$:

Stability Theorem $\Rightarrow d^\infty_B(Dg f, Dg \hat{f}) \leq \|f - \hat{f}\|_\infty$. 
More precisely...

- Density estimator $\hat{f}$ defines an order on the point cloud
  
  (sort data points by decreasing estimated density values)
More precisely...

- Density estimator \( \hat{f} \) defines an order on the point cloud (sort data points by **decreasing** estimated density values)

- Extend order to the graph edges \( \rightarrow \) **upper-star filtration**

\[
(\hat{f}([u, v]) = \min\{\hat{f}(u), \hat{f}(v)\})
\]
More precisely...

- Density estimator $\hat{f}$ defines an order on the point cloud
  (sort data points by \textbf{decreasing} estimated density values)

- Extend order to the graph edges $\rightarrow$ \textit{upper-star filtration}
  
  \[
  \hat{f}([u, v]) = \min\{\hat{f}(u), \hat{f}(v)\}
  \]

- Compute the 0-dimensional persistence diagram of this filtration
  (apply 0-dimensional persistence algorithm $\rightarrow$ union-find data structure)
Estimating the Correct Number of Clusters

Each point in the diagram represents a peak of the density in the neighborhood graph, and the vertical distance of the point to the diagonal gives the prominence of the peak.
These peaks are born from the noise in the estimator plus the use of a neighborhood graph. Their prominences are small and so they are identified as topological noise in the PD.

Estimating the Correct Number of Clusters
These peaks are born from the disconnectness of the neighborhood graph in low-density areas. They have small heights ... their clusters last forever as independent connected components. They are identified as background noise in the diagram.

Estimating the Correct Number of Clusters
These peaks correspond to the peaks of the underlying density function, even though they may not lie at the same locations in space.

Estimating the Correct Number of Clusters

6 prominent peaks
Any prominence threshold $\tau$ within the range of the prominence gap will separate the relevant peaks from the topological and background noise.
Estimating the Correct Number of Clusters

Hypotheses:

- \( f : \mathbb{R}^d \to \mathbb{R} \) a \( c \)-Lipschitz probability density function,
- \( P \subset \mathbb{R}^d \) a finite set of \( n \) points sampled i.i.d. according to \( f \),
- \( \hat{f} : P \to \mathbb{R} \) a density estimator such that \( \eta := \max_{p \in P} |\hat{f}(p) - f(p)| < \Pi/5 \),
- \( G = (P, E) \) the \( \delta \)-neighborhood graph for some positive \( \delta < \frac{\Pi - 5 \eta}{5c} \).

Note: \( \Pi \) is the prominence of the least prominent peak of \( f \).
Estimating the Correct Number of Clusters

Hypotheses:

- \( f : \mathbb{R}^d \to \mathbb{R} \) a \( c \)-Lipschitz probability density function,
- \( P \subset \mathbb{R}^d \) a finite set of \( n \) points sampled i.i.d. according to \( f \),
- \( \hat{f} : P \to \mathbb{R} \) a density estimator such that \( \eta := \max_{p \in P} |\hat{f}(p) - f(p)| < \Pi/5 \),
- \( G = (P, E) \) the \( \delta \)-neighborhood graph for some positive \( \delta < \frac{\Pi - 5\eta}{5c} \).

Note: \( \Pi \) is the prominence of the least prominent peak of \( f \)

Conclusion:

For any choice of \( \tau \) such that \( 2(c\delta + \eta) < \tau < \Pi - 3(c\delta + \eta) \), the number of clusters computed by the algorithm is equal to the number of peaks of \( f \) with probability at least \( 1 - e^{-\Omega(n)} \).

\( (the \ \Omega \ notation \ hides \ factors \ depending \ on \ c, \ \delta) \)
Estimating the Correct Number of Clusters

For any choice of $\tau$ such that $2(c\delta + \eta) < \tau < \Pi - 3(c\delta + \eta)$, the number of clusters computed by the algorithm is equal to the number of peaks of $f$ with probability at least $1 - e^{-\Omega(n)}$.

(The $\Omega$ notation hides factors depending on $c, \delta$)

Conclusion:
Estimating the Correct Number of Clusters

Proof’s main ingredient: stability theorem for persistence diagrams
Merging Clusters

- degree-0 persistence algo. builds a hierarchy of the peaks of \( \hat{f} \) (merge tree)
- merge clusters according to the hierarchy (merge each cluster into its parent)
Merging Clusters

- degree-0 persistence algo. builds a hierarchy of the peaks of $\hat{f}$ (merge tree)
- merge clusters according to the hierarchy (merge each cluster into its parent)
- given a fixed threshold $\tau \geq 0$, only merge those clusters of prominence $< \tau$

$0 \leq \tau \leq \alpha - \beta$
Merging Clusters

- degree-0 persistence algo. builds a hierarchy of the peaks of $\hat{f}$ (merge tree)
- merge clusters according to the hierarchy (merge each cluster into its parent)
- given a fixed threshold $\tau \geq 0$, only merge those clusters of prominence $< \tau$

$$\alpha - \beta < \tau \leq \gamma - \delta$$
Merging Clusters

- degree-0 persistence algo. builds a hierarchy of the peaks of $\hat{f}$ (merge tree)
- merge clusters according to the hierarchy (merge each cluster into its parent)
- given a fixed threshold $\tau \geq 0$, only merge those clusters of prominence $< \tau$

$\gamma - \delta < \tau \leq +\infty$
Pseudo-code:

**Input:** simple graph $G$ with $n$ vertices, $n$-dimensional vector $\hat{f}$, real parameter $\tau \geq 0$.

Sort the vertex indices $\{1, 2, \cdots, n\}$ so that $\hat{f}(1) \geq \hat{f}(2) \geq \cdots \geq \hat{f}(n)$;

Initialize a union-find data structure $\mathcal{U}$ and two vectors $g, r$ of size $n$;

**for** $i = 1$ to $n$ **do**

Let $\mathcal{N}$ be the set of neighbors of $i$ in $G$ that have indices lower than $i$;

**if** $\mathcal{N} = \emptyset$ // vertex $i$ is a peak of $\hat{f}$ within $G$

Create a new entry $e$ in $\mathcal{U}$ and attach vertex $i$ to it;

$r(e) \leftarrow i$ // $r(e)$ stores the root vertex associated with the entry $e$

**else** // vertex $i$ is not a peak of $\hat{f}$ within $G$

$g(i) \leftarrow \arg\max_{j \in \mathcal{N}} \hat{f}(j)$ // $g(i)$ stores the approximate gradient at vertex $i$

$e_i \leftarrow \mathcal{U}.$find($g(i)$);

Attach vertex $i$ to the entry $e_i$;

**for** $j \in \mathcal{N}$ **do**

$e \leftarrow \mathcal{U}.$find($j$);

**if** $e \neq e_i$ and $\min\{\hat{f}(r(e)), \hat{f}(r(e_i))\} < \hat{f}(i) + \tau$

$\mathcal{U}.$union($e, e_i$);

$r(e \cup e_i) \leftarrow \arg\max\{r(e), r(e_i)\} \hat{f}$;

$e_i \leftarrow e \cup e_i$;

**Output:** the collection of entries $e$ of $\mathcal{U}$ such that $\hat{f}(r(e)) \geq \tau$. 

---

**graph-based hill-climbing (1976)**

**cluster merges with persistence (2013)**
Complexity of the Algorithm

Given a neighborhood graph with $n$ vertices (with density values) and $m$ edges:

1. the algorithm sorts the vertices by decreasing density values,

2. the algorithm makes a single pass through the vertex set, creating the spanning forest and merging clusters on the fly using a union-find data structure.

→ Running time: $O(n \log n + (n + m)\alpha(n))$

→ Space complexity: $O(n + m)$

→ Main memory usage: $O(n)$
Experimental Results

Synthetic Data

Spectral clustering
\((k\text{-means in eigenspace})\)
We first run the algorithm with an arbitrary value for $\tau = 0$, and we look at the output PD.

**Experimental Results**

**Synthetic Data**

$\tau = 0$

ToMATo
Experimental Results

Synthetic Data

ToMATo
The trend of having two prominent peaks and topological noise is amplified when the number of data increases from 20k to 100k. We used the Delaunay graph as neighborhood graph, to reduce the size and speed up the computation.

Experimental Results

Synthetic Data
Experimental Results

Biological Data

Alanine-Dipeptide conformations ($\mathbb{R}^{21}$)

RMSD distance (non-Euclidean)

Common belief: 6 metastable states

PD shows anywhere between 4 and 7 clusters
Experimental Results

Biological Data

Alanine-Dipeptide conformations ($\mathbb{R}^{21}$)

RMSD distance (non-Euclidean)

Common belief: 6 metastable states
PD shows anywhere between 4 and 7 clusters
Measures of metastability confirm this insight
Experimental Results

Biological Data

Alanine-Dipeptide conformations ($\mathbb{R}^{21}$)

RMSD distance (non-Euclidean)

Note: Spectral Clustering takes a week of tweaking, while ToMATo runs out-of-the-box in a few minutes.

Experimental Results

Image Segmentation

Density is estimated in 3D color space (Luv)

Neighborhood graph is built in image domain

Distribution of prominences does not usually show a clear unique gap

Still, relationship between choice of $\tau$ and number of obtained clusters remains explicit
Recap’

ToMATo:

1. graph-based mode-seeking algorithm of [KNF’76]
2. single-pass cluster merging phase guided by persistence

Competitors:

1. Mean-Shift and its variants (smoothing a priori)
2. ...
Recap’

• Highly generic
  - applicable in arbitrary metric spaces
  - agnostic to the choice of neighborhood graph and density estimator

• Easy to tune
  - mostly two parameters: neighborhood size, persistence threshold $\tau$
  - PD provides insight into the correct number of clusters

• Comes with theoretical guarantees
  - number of obtained clusters versus number of prominent peaks
  - partial approximation of the basins of attraction of the peaks

• Efficient and practical
  - near linear runtime, linear main memory usage
  - can handle data sets with hundreds of thousands of points in practice