Cluster Analysis and other unsupervised learning methods

Nonparametric density clustering

Werner Stuetzle
Department of Statistics
University of Washington

March 27, 2017

```
## Preamble

stat593.dir <- "/Users/wxs/Dropbox/Unsupervised-learning-spring-2016"
dir.sep <- "/"

data.dir <- paste(stat593.dir, "Data", sep = dir.sep)
tc.dir <- paste(data.dir, "Test-collection", sep = dir.sep)

echo <- F
## quartz()

source(paste(stat593.dir, "Code", "gsl-functions-5-28-2010.R",
             sep = dir.sep), echo = echo)

source(paste(data.dir, "generate-artificial-data-functions-3-19-2014.R",
             sep = dir.sep), echo = echo)

library(MASS)
library(cluster)
library(colorspace)
library(mvtnorm)
## library(mclust)

options(expressions = 10000)
```

```
opts_chunk$set(fig.width=5, fig.height=5)
```

Problem summary

Given: Observations $\mathbf{x}_1, \ldots, \mathbf{x}_n$ i.i.d $\sim p(\mathbf{x})$

Goal:

- Detect presence of distinct goups
- Assign group labels to the observations

Definition of "distinct groups": Contiguous, densely populated areas of feature space, separated by contiguous, relatively empty regions.

Premise: Groups correspond to modes of $p(\mathbf{x})$

Clustering problem:

- Estimate the modes of $p(\mathbf{x})$
- Partition the feature space into "domains of attraction" (DoA's) of the modes

Questions:

- What exactly do we mean by "mode" and "DoA"?
- How do we estimate them?

Modes and DoA's for Morse densities

$p(\mathbf{x})$ = smooth density on R^m

$Dp(\mathbf{x})$ = gradient of $p(\mathbf{x})$

$Hp(\mathbf{x})$ = Hessian of $p(\mathbf{x})$ ($m \times m$ matrix of second partial derivatives).

$\mathbf{x}$ is called a critical point of p if $Dp(\mathbf{x}) = \mathbf{0}$.

$\mathbf{x}$ is called non-degenerate if $\text{rank}(Hp(\mathbf{x})) = m$

$p(\mathbf{x})$ is called a Morse density if all its critical points are non-degenerate

Vaguely speaking, a Morse density has no flat spots.

In the following, let $p(\mathbf{x})$ be a Morse density.

Definition of mode

$\mathbf{m}$ is called a mode of $p(\mathbf{x})$ if $Dp(\mathbf{m}) = \mathbf{0}$ and $Hp(\mathbf{m})$ is negative definite.

Definition of DoA

$\mathbf{m}_1, \mathbf{m}_2, \ldots =$ modes of $p(\mathbf{x})$

To determine the DoA for a point $\mathbf{x}_0$, "walk uphill" from $\mathbf{x}_0$, eventually reaching a limit point $\mathbf{x}_\infty$ with $Dp(\mathbf{x}_\infty) = \mathbf{0}$.

Prop

For almost all starting points $\mathbf{x}_0$ the limit point $\mathbf{x}_\infty$ is a mode.

If that mode is $\mathbf{m}_i$ then $\mathbf{x}_0$ is in the DoA of m_i

Note

Definition of DoA not applicable in some situations where presence of groups is obvious (for example piecewise constant densities)

This is only a problem for theory. There is no way we can decide whether a density $p(\mathbf{x})$ is Morse based on a finite sample.

Finding modes and their DoAs by optimization

Given: $\mathrm{X} = \mathbf{x}_1, \ldots \mathbf{x}_n \sim p(\mathbf{x})$ and query point $\mathbf{x}$

Goal: Assign $\mathbf{x}$ to the domain of attraction of a mode of $p(\mathbf{x})$

Old idea: Mean shift algorithms (sketch)

1. Start at $\mathbf{x}^0 = \mathbf{x}$
2. Set $\mathbf{x}^{i+1} =$ (weighted) average of k nearest neighbors of $\mathbf{x}^i$
3. Repeat (2) until convergence

(Many variations and refinements)

Motivation

The mean of the k nearest neighbor of $\mathbf{x}^i$ will be density-uphill from $\mathbf{x}^i$.

To cluster $\mathcal{X}$

- Run mean shift starting at each of the x_i and obtain limits $\mathbf{z}_i$
- Cluster the $\mathbf{z}_i$ (easy?) and obtain clusters $C_1, \ldots, C_k$
- Estimate $\mathbf{m}_u$ by mean of C_u
- Assign label u to $\mathbf{x}_i$ if $\mathbf{z}_i \in C_u$

Alternative: Plug-in estimate

- Compute density estimate $\hat{p}(\mathbf{x})$ for unknown density $p(\mathbf{x})$
- Apply numerical optimizer to $\hat{p}(\mathbf{x})$ with starting point $\mathbf{x}_i$, obtain local optimuum $\mathbf{z}_i$
- Proceed as above

Experiments with mode finding via optimization

```
## Simplex data
## ------------
##
## Spherical Gaussian point clouds with sd = 0.25 at the vertices of the
## standard simplex in m dimensions. Group sizes are 50, 60, 70,......
##
data.name <- "simplex-3"
data.filename <- paste(data.name, ".R", sep = "")
data.pathname <- paste(tc.dir, data.filename, sep = dir.sep)
source(data.pathname, echo = echo)
n <- nrow(X)
m <- ncol(X)
##
## Make kernel density estimate
##
X <- sphere(X)
cv.search.out <- cv.search(X)
h <- cv.search.out$opt.smopar
h
```

```
## [1] 0.3648376

density <- make.gaussian.kernel.density.estimate(X, h)
#
opt.density <- function(x) {
  -density(matrix(x, nrow = 1))
}
##
control <- list("maxit" = 500)
optim.out.list <- NULL
##
for (i in 1:n) {
  optim.out <- optim(t(as.matrix(X[i,])), opt.density, method = "BFGS",
                     control = control)
  if (is.null(optim.out.list)) optim.out.list <- list(optim.out)
  else optim.out.list <- c(optim.out.list, list(optim.out))
}
##
convergence <- rep(0, n)
optima <- matrix(0, nrow = n, ncol = m)
for (i in 1:n) {
  convergence[i] <- optim.out.list[[i]]$convergence
  optima[i,] <- optim.out.list[[i]]$par
}
sum(convergence)

## [1] 0

## [1] 0
## All 180 optimization runs converged.
## Cluster local optima using Ward's method
##
hclust.out <- hclust(dist(optima), method = "ward.D")
rev(hclust.out$height)[1:20]

##  [1] 1.335895e+02 9.895631e+01 3.252400e+01 2.323216e+01 1.106348e+01
##  [6] 2.822690e+00 3.457262e-03 1.635070e-03 1.317575e-03 7.141366e-04
## [11] 6.412651e-04 6.405853e-04 6.302945e-04 5.475859e-04 5.443208e-04
## [16] 4.074242e-04 3.944519e-04 3.722461e-04 3.652680e-04 3.634001e-04

##
## Clear gap between 6 and 7, so there appear to be 7 local optima
##
cluster.labels <- cutree(hclust.out, k = 7)
table(group.id, cluster.labels)
```

```
##          cluster.labels
## group.id  1  2  3  4  5  6  7
##        1 31 17  2  0  0  0  0
##        2  0  0  0 39 19  2  0
##        3  0  0  0  0  0 69  1
```

```
##
## Procedure splits groups 1 and 2. Not so encouraging.
##
##=====================================================================
##
## Same for olive oil data
## Read pre-computed density optima. Finding the optima takes about 15min.
##
data.name <- "olive"
density.optima.filename <- paste(data.name, "-density-optima.R", sep = "")
density.optima.pathname <- paste(data.dir, density.optima.filename,
                                 sep = dir.sep)
source(density.optima.pathname)
##
n <- 572
m <- 8
convergence <- rep(0, n)
optima <- matrix(0, nrow = n, ncol = m)
for (i in 1:n) {
  convergence[i] <- optim.out.list[[i]]$convergence
  optima[i,] <- optim.out.list[[i]]$par
}
sum(convergence)
```

```
## [1] 0
```

```
##
## All optimizations converged
##
## Cluster local optima using Ward's method
##
hclust.out <- hclust(dist(optima), method = "ward.D")
rev(hclust.out$height)[1:20]
```

```
##  [1] 150.23661 145.64250 105.59982  91.56043  67.55352  57.49348  39.65697
##  [8]  38.70020  37.28883  29.64935  29.45185  28.13144  27.55466  26.92762
## [15]  25.54084  24.87391  24.81490  24.37030  22.53394  21.76417
```

```
rev(hclust.out$height)[490:520]
```

```
##  [1] 5.591351e-01 5.569003e-01 5.443401e-01 5.237513e-01 5.079627e-01
##  [6] 4.496456e-01 4.200144e-01 4.073459e-01 4.029780e-01 3.706531e-01
## [11] 2.447035e-01 1.389417e-03 5.680384e-04 2.446552e-04 1.387726e-04
## [16] 1.146709e-04 9.451924e-05 7.986208e-05 7.636761e-05 6.950371e-05
## [21] 6.877644e-05 6.824008e-05 5.719147e-05 4.564141e-05 3.659164e-05
## [26] 3.514780e-05 3.481700e-05 3.084302e-05 3.008544e-05 3.006295e-05
## [31] 2.753799e-05

##
## Big jump between heights 500 and 501. Looks like there are 501 modes!!!!
##
```

Problems with mode finding via optimization

- Density estimates are noisy (as all estimates)
- Some modes of a density estimate $\hat{p}$ will reflect modes of the underlying density p while others will be due to sampling variability
- Need a way to assess the "prominence" of modes and eliminate clusters corresponding to modes with low prominence.
- Hard to do that using the optimization approach to mode finding - have not seen any proposals

Will now introduce an alternative approach to nonparametric density clustering that is based on analysis of level sets.

This approach

- Leads to a definition of "modes" that is not based on derivatives
- Offers several ways of measuring the prominence of modes
- Offers ways of assessing the significance of modes ("Clustering with confidence")

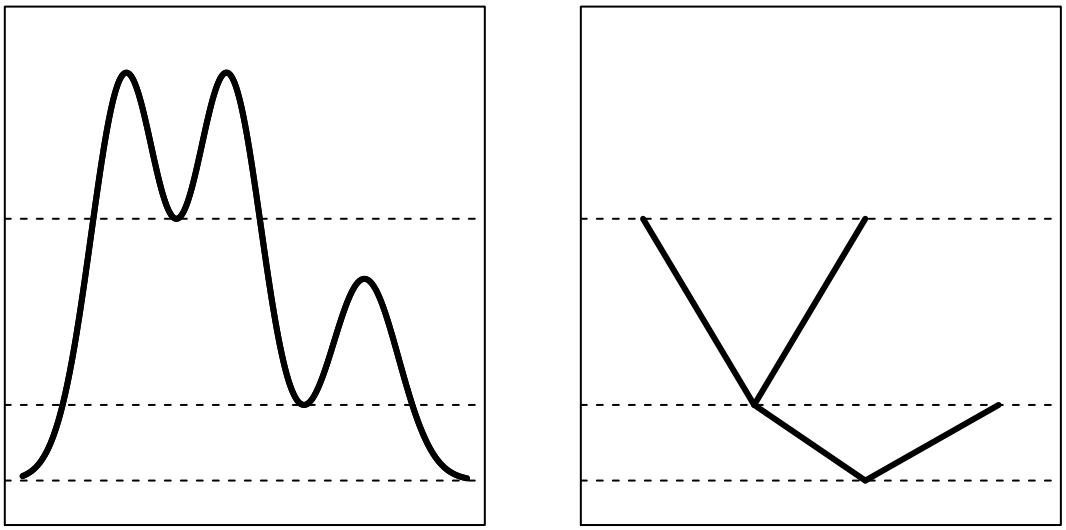
The cluster tree of a density

Define the level set $L(\lambda; p)$ of density p at level λ as the subset of feature space for which the density exceeds λ (Hartigan 1975):

$$L(\lambda; p) = \{\mathbf{x} \,|\, p(\mathbf{x}) > \lambda\}\,.$$

Hartigan pointed out that connected components of level sets have a tree structure:
For any two components A, B (possibly at different levels) we have $A \subset B$, or $B \subset A$, or $A \cap B = \emptyset$.

Cluster tree is easiest to define recursively:



Each node N of cluster tree

- represents a subset D(N) of feature space (high density cluster)
- is associated with a density level $\lambda(N)$

The root node

- represents the entire support of the density;
- is associated with density level $\lambda(N) = 0$.

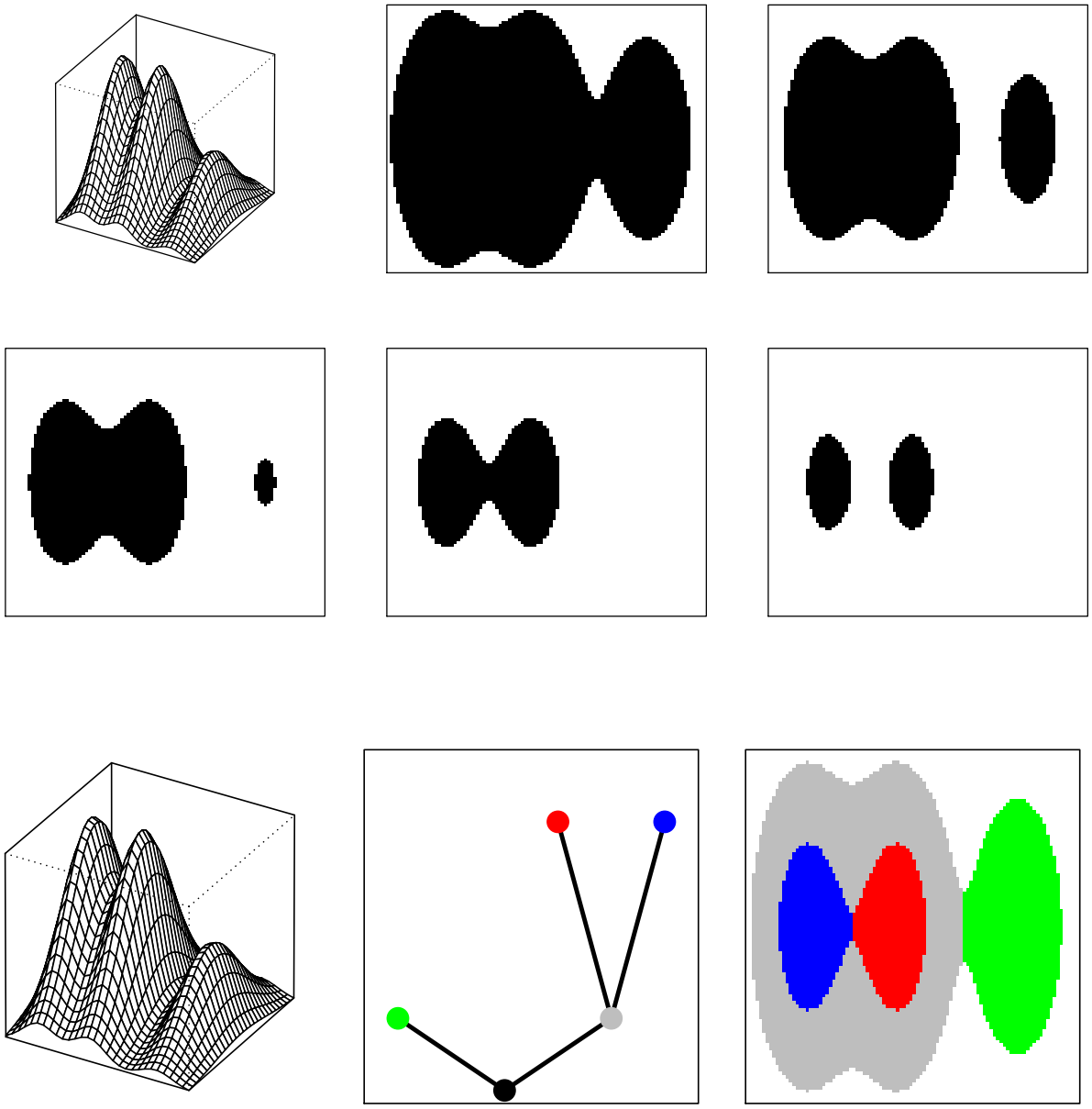
To determine descendents of node N

- Find lowest level λ_d for which $L(\lambda; p) \cap D(N)$ has two connected components
- If there is no such λ_d then N is leaf of the tree
- Otherwise, create daughter nodes representing the connected components, with associated level λ_d, and recurse.

Note

- Level sets with multiple connected components indicate multi-modality.
- If the density is Morse, the leaves of the tree correspond to the modes of the density
- There might not be a single level for which level set reveals all the modes.

A 2d illustration



Estimating the cluster tree of a density

Obvious approach: Plug-in estimate

- Estimate p by (nonparametric) density estimate $\hat{p}$;
- Estimate cluster tree of p by cluster tree of $\hat{p}$.

However, there are computational as well as statistical problems:

- How do we compute (or approximate) level sets and their connected components?
- How do we distinguish spurious components (modes) due to sampling variability from real components reflecting the structure of the true density?

Will first talk about computational issues

Computing the cluster tree of a piecewise constant density estimate

Suppose that density estimate $\hat{p}$ is piecewise constant over disjoint (hyper-) rectangles forming a partition of feature space:

$$\hat{p}(\mathbf{x}) = \sum_{i=1}^{m} c_i \, I(\mathbf{x} \in R_i) \, .$$

Example: Histograms, ASH estimates, Cart estimates, kernel estimates with hyper-rectangular kernels (careful!).

In this case, level sets, their connected components, and the cluster tree can be computed exactly.

Basic idea: Convert geometry problem into graph problem.

Define edge weighted graph $G = (V, E, W)$:

- Vertices represent rectangles R_i;
- Edges encode adjacency: (i, j) is an edge if R_i is adjacent to R_j;
- Weight of vertex i is value of density in rectangle i: $w_i = c_i$;
- Weight of edge (i, j) is minimum of density in the two (adjacent) rectangles: $w_{ij} = \min(c_i, c_j)$.

Define threshold graph $G^-(\lambda)$:

- Remove all vertices with $w_i \leq \lambda$;

- Remove all edges with $w_{ij} \leq \lambda$.

Connected components of level set $L(\lambda; \hat{p})$ correspond to connected components of threshold graph $G^-(\lambda)$.

Finding connected components of $G^-(\lambda)$ is a standard graph problem.

1. Start graph traversal at an arbitrary vertex and mark all visited vertices;
2. Remove visited vertices and their incident edges;
3. Repeat (1) and (2) until no more vertices remain.

The cluster tree can be computed recursively:

- Each node represents a sub-graph of G and the corresponding subset of feature space;
- The root node represents the entire graph G and the support of $\hat{p}$;
- To find the descendents of a node N representing a graph H we find the smallest value of λ for which $H^-(\lambda)$ has two connected components;
- If there is no such λ then N is a leaf of the tree;
- Otherwise we create daughter nodes representing the connected components of $H^-(\lambda)$ and recurse

---

Computing the cluster tree of the 1-nn density estimate

Recall definition of k-nn density estimate

$S(\mathbf{x}, r)$: Sphere around query point $\mathbf{x}$ with radius r.
k : Number of observations in $S(\mathbf{x}, r)$
V_m: Volume of unit sphere in R^m

$$\begin{aligned} \mathbf{E}(k) &= n \int_{S(\mathbf{x},r)} p(\mathbf{y})\, \mathrm{d}\mathbf{y} \\ &\approx n\, r^m\, V_m\, p(\mathbf{x}) \end{aligned}$$

Replace $\mathbf{E}(k)$ by its observed value $\Rightarrow$

$$\hat{p}(\mathbf{x}) = \frac{k}{n\, r^m\, V_m}$$

k-nn density estimate: Fix k and observe r.
r = radius of smallest sphere around $\mathbf{x}$ containing k observations.

1-nn density estimate:

$$\hat{p}_1(\mathbf{x}) = \frac{1}{n\, r^m\, V_m}$$

where

$$r = d(\mathbf{x}, \mathcal{X}) = \min(d(\mathbf{x}, \mathbf{y} \mid \mathbf{y} \in \mathcal{X}))$$

Level sets of the 1-nn density estimate

Define

$$r(\lambda) = (n\, \lambda^m\, V_m)^{-\frac{1}{m}}$$

The level set $L(\lambda; \hat{p}_1)$ is the union of open spheres around the observations with radius $r(\lambda)$.

G = complete graph over $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with edge weights $w_{i,j} = d(\mathbf{x}_i, \mathbf{x}_j)$.

Prop

The connected components of $L(\lambda; \hat{p}_1)$ are "the same" as the connected components of $G^+(2r(\lambda))$

We already know that the connected components of $G^+(\mu)$ are the same as the connected components of $T^+(\mu)$ where T is the MST of G.

Therefore, we can compute cluster tree of nearest neighbor density estimate by

- Breaking longest MST edge, thereby splitting MST into two subtrees
- Recursively applying splitting process to subtrees

Prop

The cluster tree of the 1-nn density estimate is isomorphic to the single linkage dendogram.

Problem

- The 1-nn density estimate is noisy and has a singularity at every data point $\Rightarrow$
- The cluster tree of the 1-nn density estimate is a poor estimate of the cluster tree of the underlying density
- It needs to be pruned