Probability density functions on star domains with an application to classification

Claudia Fassino[1]

joint work with
Carlos Cuevas-Covarrubias[2], Eva Riccomagno[1]
and
Carmen Villar-Patiño[2]

[1] Dipartimento di Matematica, Università di Genova, Italy
[2] Facultad de Ciencias Actuariales, Universidad Anáhuac, México

Introduction

- This talk is about a construction of probability density functions supported on closed and bounded star domains.

Introduction

- This talk is about a construction of probability density functions supported on closed and bounded star domains.

Figure: A “natural” closed and bounded star domain: the starfruit.

Introduction

- This talk is about a construction of probability density functions supported on closed and bounded star domains.

Figure: A "natural" closed and bounded star domain: the starfruit.

- The construction exploits tools from numerical computational algebra.

Probability density functions on bounded star domains

The density function defined here is represented by three equations.

Let $\Omega \subset \mathbb{R}^n$ be a bounded star domain s.t. its frontier is described by $f(x_0) = 0$, with f polynomial, and let g be a non negative continuous function. Let Σ be a surface in $\mathbb{R}^{n+1}$ defined by

$$\begin{cases} x = z + s(x_0 - z) \\ f(x_0) = 0 \\ x_{n+1} = g(s) \end{cases} \qquad \text{with } s \in [0,1] \tag{1}$$

The density function defined here is represented by three equations.

Let $\Omega \subset \mathbb{R}^n$ be a bounded star domain s.t. its frontier is described by $f(x_0) = 0$, with f polynomial, and let g be a non negative continuous function. Let Σ be a surface in $\mathbb{R}^{n+1}$ defined by

$$\begin{cases} x = z + s(x_0 - z) \\ f(x_0) = 0 \\ x_{n+1} = g(s) \end{cases} \qquad \text{with } s \in [0,1] \tag{1}$$

If the surface Σ and the star domain Ω bound a region $D \subset \mathbb{R}^{n+1}$ s.t. $\mathrm{Vol}(D) = 1$, then (1) defines a probability density function.

Star domains

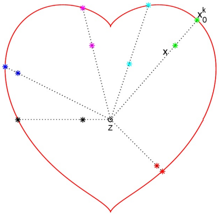
- Let $\Omega \subset \mathbb{R}^n$ be a star domain, that is there exists (a vantage point) $z \in \Omega$ s.t. $\forall x \in \Omega$ the segment from z to x is contained in Ω.
- Let f be a polynomial s.t. $f(x_0) = 0$ bounds the star domain Ω.

The parametric description of Ω w.r.t. the vantage point z is given by

$$\begin{cases} x = z + s(x_0 - z) \\ f(x_0) = 0 \end{cases} \qquad \text{with } s \in [0, 1]$$

Example: Star-shaped set with boundary given by

$$\{(x_1, x_2) \in \mathbb{R}^2 : (x_1^2 + x_2^2 - 1)^3 - x_1^2 x_2^3 = 0\}$$



Choice of the probability density function g

Probability density function:

$$\begin{cases} x = z + s(x_0 - z) \\ f(x_0) = 0 \\ x_{n+1} = g(s) \end{cases} \qquad \text{with } s \in [0,1]$$

The function g is strictly decreasing in $[0,1]$, with $g(1) = 0$, highest value $g(0)$ and the level curves of Σ are contractions of Ω.

Choice of the probability density function g

Probability density function:

$$\begin{cases} x = z + s(x_0 - z) \\ f(x_0) = 0 \\ x_{n+1} = g(s) \end{cases} \qquad \text{with } s \in [0,1]$$

The function g is strictly decreasing in $[0,1]$, with $g(1) = 0$, highest value $g(0)$ and the level curves of Σ are contractions of Ω.

- Cone: $g_c(s) = \dfrac{n+1}{\mathrm{Vol}(\Omega)}(1-s)$.
- Paraboloid: $g_p(s) = \dfrac{n+2}{2\,\mathrm{Vol}(\Omega)}(1-s^2)$.
- Ellipsoid: $g_e(s) = \dfrac{2}{\mathrm{B}\left(\frac{n}{2}+1, \frac{1}{2}\right)\,\mathrm{Vol}(\Omega)}\sqrt{1-s^2}$.

Mixture of probability density functions

Starting from g_c, g_p and g_e a new probability density function supported on Ω is

$$\begin{cases} x = z + s(x_0 - z) \\ f(x_0) = 0 \\ x_{n+1} = \alpha g_c(s) + \beta g_p(s) + (1 - \alpha - \beta) g_e(s) \end{cases} \qquad s \in [0, 1]$$

where $\alpha, \beta \in [0, 1]$ and $\alpha + \beta \leq 1$.

The x_{n+1} component is a decreasing function of $s \in [0, 1]$.

PolyStar classification algorithm

Given a finite set of labelled points in $\mathbb{R}^n$ partitioned in k categories, the classification PolyStar algorithm consists of two parts:

PolyStar classification algorithm

Given a finite set of labelled points in $\mathbb{R}^n$ partitioned in k categories, the classification PolyStar algorithm consists of two parts:

- model construction: for each $j = 1, \ldots, k$ we construct the system

$$\begin{cases} x = z_j + s_j(x_0 - z_j) \\ f_j(x_0) = 0 \\ x_{n+1} = g_j(s_j) \end{cases} \qquad \text{with } s_j \in [0, 1]$$

PolyStar classification algorithm

Given a finite set of labelled points in $\mathbb{R}^n$ partitioned in k categories, the classification PolyStar algorithm consists of two parts:

- model construction: for each $j = 1, \ldots, k$ we construct the system

$$\begin{cases} x = z_j + s_j(x_0 - z_j) \\ f_j(x_0) = 0 \\ x_{n+1} = g_j(s_j) \end{cases} \qquad \text{with } s_j \in [0, 1]$$

- point classification: the point x is
  - allocated to the category j for which $g_j(s_j(x))$ is largest and positive and
  - not classified if there are ties or all g_j are negative or complex numbers.

PolyStar classification algorithm

Given a finite set of labelled points in $\mathbb{R}^n$ partitioned in k categories, the classification PolyStar algorithm consists of two parts:

- model construction: for each $j = 1, \ldots, k$ we construct the system

$$\begin{cases} x = z_j + s_j(x_0 - z_j) \\ f_j(x_0) = 0 \\ x_{n+1} = g_j(s_j) \end{cases} \qquad \text{with } s_j \in [0, 1]$$

- point classification: the point x is
  - allocated to the category j for which $g_j(s_j(x))$ is largest and positive and
  - not classified if there are ties or all g_j are negative or complex numbers.

Different allocation criteria: x is assigned to the cluster j s.t.

$$\frac{w_j g_j(s(x))}{\sum_{j=1}^{k} w_j g_j(s(x))} \quad \text{is largest.}$$

Model construction

For each j, the construction of the system

$$\begin{cases} x = z_j + s_j(x_0 - z_j) \\ f_j(x_0) = 0 \\ x_{n+1} = g_j(s_j) \end{cases} \qquad \text{with } s_j \in [0, 1]$$

requires

Model construction

For each j, the construction of the system

$$\begin{cases} x = z_j + s_j(x_0 - z_j) \\ f_j(x_0) = 0 \\ x_{n+1} = g_j(s_j) \end{cases} \qquad \text{with } s_j \in [0,1]$$

requires

- a boundary polynomial f_j for Ω_j (a-priori known or estimated processing a finite set of boundary points by NBM, LDP...),

Model construction

For each j, the construction of the system

$$\begin{cases} x = z_j + s_j(x_0 - z_j) \\ f_j(x_0) = 0 \\ x_{n+1} = g_j(s_j) \end{cases} \qquad \text{with } s_j \in [0,1]$$

requires

- a boundary polynomial f_j for Ω_j (a-priori known or estimated processing a finite set of boundary points by NBM, LDP...),
- a (given or estimated) vantage point z_j,

Model construction

For each j, the construction of the system

$$\begin{cases} x = z_j + s_j(x_0 - z_j) \\ f_j(x_0) = 0 \\ x_{n+1} = g_j(s_j) \end{cases} \qquad \text{with } s_j \in [0,1]$$

requires

- a boundary polynomial f_j for Ω_j (a-priori known or estimated processing a finite set of boundary points by NBM, LDP...),
- a (given or estimated) vantage point z_j,
- a probability density function g_j and
- the computation of $\mathrm{Vol}(\Omega_j)$ (invariant by the choice of z_j) for normalising g_j.

The performance of PolyStar will be strongly effected by the choice of the vantage point.

Point classification

$$\begin{cases} x = z_j + s_j(x_0 - z_j) \\ f_j(x_0) = 0 \\ x_{n+1} = g_j(s_j) \end{cases} \qquad \text{with } s_j \in [0,1]$$

The classification of a single point $x \in \mathbb{R}^n$ requires, for each $j = 1, \dots, k$,

- the computation of $s_j(x)$ and
- the evaluation of the univariate function $g_j(s_j(x))$.

The value $s_j(x)$ is the Minkowski functional of x w.r.t. Ω_j.
Since $x_0 = z_j + (x - z_j)/s_j(x)$, we have

$$f_j(x_0) = 0 \quad \Leftrightarrow \quad f_j(z_j + (x - z_j)/s_j(x)) = 0$$

and so we compute $s_j(x)$ applying a root finding method to the univariate equation $f_j(z_j + (x - z_j)/s_j(x)) = 0$.

Computational cost

- The construction of the model is done once.
  For each j, the computational cost is close to $O(\#I_j^2)$, for low degree polynomial computed using the NBM algorithm.

Computational cost

- The construction of the model is done once.
  For each j, the computational cost is close to $O(\# I_j^2)$, for low degree polynomial computed using the NBM algorithm.
- The classification of a point $x \in \mathbb{R}^n$ requires, for each j, the computation of $s_j(x)$ and of $g_j(s_j(x))$.
  The computational cost of a root finding (as the Newton method) applied to $f_j(z_j + (x - z_j)/s_j(x)) = 0$ is $O(d_j)$ where d_j is the total degree of f_j. The computational cost for $g_j(s_j(x)$ is linear in the degree of g_j.
  If g_j are low degree polynomials, then the computational cost for the classification of a point x is $kO(\widehat{d})$ where $\widehat{d} = max_{j \in \{1,\dots,k\}}\{d_j\}$.

Calibration

At times it might be needed a dilation of Ω, for instance for reducing the number of non classified points.

Given $\varepsilon > 0$ and $z \in \mathbb{R}^n$, the dilation function $d_{\varepsilon,z}(x) = z + (1+\varepsilon)(x - z)$ defines the set

$$\boxed{d_{\varepsilon,z}(\Omega) = \{\tilde{x} = d_{\varepsilon,z}(x), x \in \Omega\} = \{(1+\varepsilon)x \, : \, x \in \Omega\} = (1+\varepsilon)\Omega}$$

It is a dilation of Ω, a star domain with vantage point z and its volume is $(1+\varepsilon)^n \, \mathrm{Vol}(\Omega)$.

Calibration

At times it might be needed a dilation of Ω, for instance for reducing the number of non classified points.

Given $\varepsilon > 0$ and $z \in \mathbb{R}^n$, the dilation function $d_{\varepsilon,z}(x) = z + (1+\varepsilon)(x - z)$ defines the set

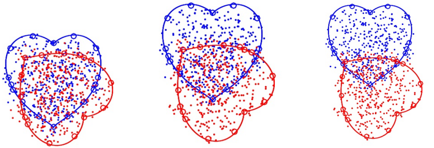
$$\boxed{d_{\varepsilon,z}(\Omega) = \{\tilde{x} = d_{\varepsilon,z}(x), x \in \Omega\} = \{(1+\varepsilon)x \,:\, x \in \Omega\} = (1+\varepsilon)\Omega}$$

It is a dilation of Ω, a star domain with vantage point z and its volume is $(1+\varepsilon)^n \,\mathrm{Vol}(\Omega)$.

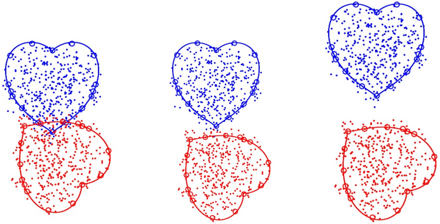
We work over $d_{\varepsilon,z}(\Omega)$ recycling the computations done for the original Ω. Starting from x, we compute the point $d_{\varepsilon,z}^{-1}(x) \in \Omega$ and its Minkowski functional s w.r.t. Ω. The probability density function over $d_{\varepsilon,z}(\Omega)$ is given by

$$\boxed{x_{n+1} = \frac{g(s)}{(1+\varepsilon)^n}}$$

A simulative example: blue and red hearts



(a) b=0 (b) b=0.5 (c) b=1



(d) b=1.5 (e) b=2 (f) b=2.5

A simulative example: blue and red hearts

- Let Ω_B be the star domain bounded by $(x_1^2 + x_2^2 - 1)^3 - x_1^2 x_2^3 = 0$ and with $z = 0$.
- $\Omega_{B,b}$ is obtained by translating Ω_B along the x_2-axis in such a way that the vantage point becomes $(0, b)$ with $b \in \{0, 0.5, 1, 1.5, 2, 2.5\}$.
- Ω_R is obtained by rotating Ω_B clockwise by $\pi/4$.

Since Ω_R and $\Omega_{B,b}$ have the same volume, for classifying a single point we only have to compare the s-values associated to each heart. It does not matter which surface g we use as long as it is the same for both clusters.

The further apart are the two hearts, the better is the classification.

A dilation $\varepsilon = 0.3$ is applied to Ω_R and $\Omega_{B,b}$ or to none. Dilation improves classification and reduces drastically the number of NC points.

A simulative example: blue and red hearts

	$\varepsilon = 0$			$\varepsilon = 0.3$		
Cluster	Exact	Wrong	NC	Exact	Wrong	NC
Ω_R	55.3	40.8	3.9	58.7	40.8	0.5
$\Omega_{B,0}$	57.2	38.1	4.7	61.1	38.9	0
Ω_R	68.4	26.5	5.1	72.9	26.6	0.5
$\Omega_{B,0.5}$	67.6	27.2	5.2	71.8	28.2	0
Ω_R	81.1	12.6	6.3	86.3	13.2	0.5
$\Omega_{B,1}$	78.9	14.5	6.6	85.1	14.7	0.2
Ω_R	86.6	4.8	8.6	93.9	5.6	0.5
$\Omega_{B,1.5}$	86.6	2.9	10.5	95.9	3.9	0.2
Ω_R	88.5	0	11.5	99.0	0	1
$\Omega_{B,2}$	87.7	0	12.3	99.3	0.5	0.2
Ω_R	88.5	0	11.5	99.0	0	1
$\Omega_{B,2.5}$	87.6	0	12.4	99.8	0	0.2

Table: Percentages of points classified correctly (Exact), attributed to the wrong set (Wrong), or not classified (NC).

Example: colours

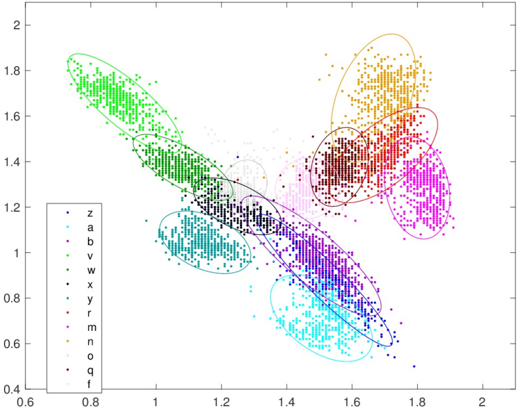


Figure: Left plot [4] shows the picture of 13 chickpeas of different colours, labelled $a, b, f, m, n, o, q, r, v, w, x, y, z$. Using the CIELAB model, 500 points of $\mathbb{R}^2$ were sampled for each colours.

Example: colours

For each j: Ω_j is bounded by an ellipsis and the density function g_j is a mixture, that is $g_j = \alpha g_c + \beta g_p + (1 - \alpha - \beta) g_e$.

	NBM			
ε	(α, β)	*SR*	*min*	*NC*
0	(0, 0.7)	83.8	64.6	3.9
0	(0.5, 0.3)	83.6	69.4	3.9
0.1	(0 , 0.9)	85.1	64.8	2.0
0.1	(0.7, 0.2)	84.9	70.4	2.0
0.2	(0.1, 0.9)	85.9	63.2	1.0
0.2	(1, 0)	85.7	70.4	1.0

For each ε the values of (α, β) are s.t. either the mean success rate (SR) or the minimum of the correct classification rates (min) are maximum.
NC depends only on the dilation parameter because the *NC* points are those outside the star domain basis.

Example: Comparison with the benchmark method k-NN

The true advantage PolyStar has over the other methods is its computational cost.

The cost of classifying a point PolyStar requires $13O(\hat{d})$ ($\hat{d} = 2$ for an elliptical basis).

The k-NN algorithm for each single point requires

- for $k = 1$: $O(2v)$ and
- for $k = 5, 10$: $O(vk)$

where v is the size of the training set.

Alg.	SR	min
1-NN	89	50
5-NN	89	70
10-NN	88	70
$\mathrm{PolyStar}_1$	85.9	63.2
$\mathrm{PolyStar}_2$	85.7	70.4

References

B. Buchberger and H. M. Möller *The construction of multivariate polynomials with preassigned zeros* EUROCAM'82, pp 24–31 (1982)

The CoCoA Team, *CoCoA: a system for doing Computations in Commutative Algebra*, available at `http://cocoa.dima.unige.it`.

C. Fassino, *Vanishing Ideal of Limited Precision Points* J. Symb. Comput. Vol 45, pp 19–37 (2010).

M. C. Villar Patiño, C. Cuevas Covarrubias, *Controlled condensation in k-NN and its application for real time color identification* Revista de Matemática: teoriá y aplicationes, 23(1) : 143–154 (2016).

C. Cuevas Covarrubias, C. Fassino, E. Riccomagno, M. C. Villar Patiño, *Probability density functions on star domains with an application to classification* In preparation, (2016)