Intrinsic dimension estimation using the Maximum Likelihood algorithm

Paper: Translated_Poisson_Mixture_Model_for_Stratification_Learning_PREPRINT

Previous Manifold Learning Assumption: All the data points being analyzed come from the same manifold. This means all the data points can be described by the same underlying geometric structure. Under this assumption, there is a single intrinsic dimension for the entire dataset.

Problem with the Assumption: Different subsets of the (noisy) data might have different underlying structures and complexities. This work focuses on detecting and dealing with these varying complexities in the same dataset.

Idea:

The study is based on stratification learning.

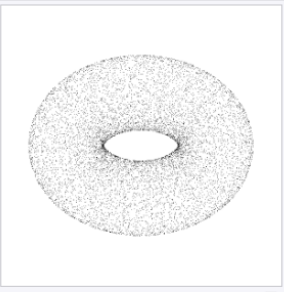
> ### Beyond manifold learning
>
> Stratification learning extends the concept of manifold learning for better handling of geometrically structured data in high dimensional spaces. Manifold learning relies on the assumption that high dimensional data in real-world applications is often sparse and distributed along with a low dimensional manifold. Stratification learning assumes that data are distributed along with a *stratified space*, which is a composite of manifolds of varying dimensions. Moreover, such data are stratified according to *strata*, i.e., component manifolds of the stratified space. By utilizing such a geometric structure, we can achieve an unprecedentedly efficient learning!

The study does not assume linear subspaces. Instead, it builds upon the previous approach of computing the intrinsic dimension at each point using a Maximum Likelihood (ML) estimator based on a Poisson distribution.

Rather than estimating the intrinsic dimension independently for each point, the approach computes a Maximum Likelihood (ML) estimation on the entire point cloud data simultaneously. This is achieved using a Translated Poisson mixture model.

This technique provides a soft clustering(each data point can belong to more than one cluster) of the data points based on dimensionality and density. It automatically estimates both the intrinsic dimension and density for each identified class. This methodology ensures robustness to noise and outliers while accurately capturing the intrinsic structure and complexity of the data.

A **point cloud** is a discrete set of data points in space. The points may represent a 3D shape or object. Each point position has its set of Cartesian coordinates (X, Y, Z).[1] Point clouds are generally produced by 3D scanners or by photogrammetry software, which measure many points on the external surfaces of objects around them. As the output of 3D scanning processes, point clouds are used for many purposes, including to create 3D computer-aided design (CAD) or geographic information systems (GIS) models for manufactured parts, for metrology and quality inspection, and for a multitude of visualizing, animating, rendering, and mass customization applications.



A point cloud image of a torus

Alignment and registration [edit]

Methodology:

Local intrinsic dimension estimation:

Levina and Bickel, [23], proposed a geometric and probabilistic method which estimates the local dimension and density of a point cloud data. This dimension estimator is equivalent to the one proposed in [32] in the context of dynamical systems. Their approach is based on the idea that if we sample an m-dimensional manifold with T points, the proportion of points that fall into a ball around a point x_t is $\frac{k}{T} \approx \rho(x_t)V(m)R_k(x_t)^m$. The given point cloud, embedded in high dimensions D, is $X = \{x_t \in \mathbb{R}^D; t = 1, \ldots, T\}$, k is the number of points inside the ball, $\rho(x_t)$ is the local sampling density at point x_t, $V(m)$ is the volume of the unit sphere in $\mathbb{R}^m$, and $R_k(x_t)$ is the Euclidean distance from x_t to its k-th nearest neighbor (kNN). Then, they consider the inhomogeneous process $N(R, x_t)$, which counts the number of points falling into a small D-dimensional sphere $B(R, x_t)$ of radius R centered at x_t. This is a binomial process, and some assumptions need to be done to proceed. First, if $T \to \infty$, $k \to \infty$, and $k/T \to 0$, then we can approximate the binomial process by a Poisson process. Second, the density $\rho(x_t)$ is considered constant inside the sphere, a valid assumption for small

R. Note that the latter assumption is only local, the global density does not need to be constant, only inside the local sphere. With these assumptions, the rate λ of the counting process $N(R, x_t)$ can be written as

$$\lambda(R, x_t) = \rho(x_t)V(m)mR^{m-1}. \tag{1}$$

The log-likelihood of the process $N(R, x_t)$ is then given by

$$L(m(x_t), \theta(x_t)) = \int_0^R \log \lambda(r, x_t) dN(r, x_t) - \int_0^R \lambda(r, x_t) dr,$$

where $\theta(x_t) := \log \rho(x_t)$ is the density parameter and the first integral is a Riemann-Stieltjes integral [29]. The maximum likelihood estimators lead to a computation for the local dimension at point x_t, $m(x_t)$, depending on all the neighbors within a distance R from x_t [23]. In practice, it is more convenient to compute a fixed amount k of nearest neighbors. Thus, the local estimators at point x_t are

$$m(x_t) = \left[\frac{1}{k-1} \sum_{j=1}^{k-1} \log \frac{R_k(x_t)}{R_j(x_t)} \right]^{-1}, \tag{2}$$

$$\theta(x_t) = \log \left((k-1) / \left(V(m(x_t)) R_k(x_t)^{m(x_t)} \right) \right), \tag{3}$$

where $V(m(x_t)) = (2\pi^{m(x_t)/2})/(m(x_t)\Gamma(\frac{m(x_t)}{2}))$, and $\Gamma(\frac{m(x_t)}{2}) = \int_0^\infty t^{m(x_t)/2-1} e^{-t} dt$. If the data points be-

Intrinsic Dimension Calculation for data points belonging to the same manifold: average over all local estimators m(xt) in order to obtain a more robust estimator.
But if there are manifolds with different dimensions, averaging is not a valid solution for finding ID.
Proposed solution for this problem:
First cluster according to dimensionality and then estimate the dimensionality for each cluster.
Another possibility is to include this in the process via the simultaneous soft clustering and estimation technique.

Noise Handling Framework Using the Translated Poisson Model

- The underlying point process, which represents the true data points, is translated to an observable point process. This translation accounts for the noise present in the data.
- The points' input and output spaces are not necessarily identical and can even have different dimensions. For example, noise can cause points to move from the underlying manifold into a higher-dimensional embedding space.
- An input point at location x in the input space X is randomly translated to a location z in the output space Z, according to a conditional probability density f(z|x), called the transition density.
- Each point is translated independently of the others and there are no deletions or insertions in the translation process.

We have the following critical theorem [30] which says that a translated Poisson process is also a Poisson process:

Theorem (Snyder & Miller [30]). *Let* $\{N(A)\colon A \subseteq X\}$ *be a Poisson process with an integrable intensity function* $\{\lambda(x)\colon x \in X\}$. *Points of this input point process are translated to the output space* Z *to form the output point process* $\{M(B)\colon B \subseteq Z\}$, *where each point is independently translated according to the transition density* $f(z|x)$. *Then, if there are no insertions and deletions,* $\{M(B)\colon B \subseteq Z\}$ *is a Poisson process with intensity*

$$\mu(z) = \int_X f(z|x)\lambda(x)dx.$$

Since the intensity of the Poisson process in our model is parametrized by the Euclidean distances of the points (and not by the points themselves, see previous Section), we are going to consider a random translation in the distances. This means that we do not observe the original distances but noisy distances. Let $f(s|r)$ be the transition density which defines the random process which translates a distance r in the input space to a distance s in the observable space. If $\lambda(r, x_t)$, defined in (1), is the local rate of the Poisson process which defines the counting process in the input space, then $\mu(s)$, the intensity of the Poisson process in the output space is given by

$$\mu(s, x_t) = \int_0^{R'} f(s|r) e^{\theta} V(m) m r^{m-1} dr. \tag{4}$$

R' is different from the radius R considered in the counting process $N(R, x_t)$. We consider $R' > R$ in (4) because, points originally at distance greater than R from x_t can be placed within a distance less than R after the translation process. In practice, the maximum translation is small (just a perturbation because of the noise) and we consider $R' = R + \sigma$ in the particular case of a Gaussian transition density (11). The log-likelihood of the translated Poisson process is

$$L(m(x_t), \theta(x_t)) = \int_0^R \log(\mu(s, x_t)) dN(s, x_t) - \int_0^R \lambda(r, x_t) dr.$$

The parameters of the maximum log-likelihood are obtained by solving the system of equations $\partial L/\partial m = 0$ and $\partial L/\partial \theta = 0$. We then obtain the following expression for m when we use the k nearest neighbors (k-NN) instead of the

points within distance less to R,

$$m(x_t)=\left[\frac{1}{k-1}\sum_{i=1}^{k-1}\frac{\int_0^{R'} f(R_i(x_t)|r)r^{m-1}\log\frac{R_k(x_t)}{r}dr}{\int_0^{R'} f(R_i(x_t)|r)r^{m-1}dr}\right]^{-1} \quad (5)$$

where, by an abuse of notation, we have identified $m = m(x_t)$ in the right hand side. Note that this expression reduces to the Levina and Bickel estimator [23] in the particular case that $f(s|r) = \delta(s-r)$, i.e., there is no translation of the original points. This corresponds to the ideal case with no noise.

Equation (5) is a nonlinear recursive expression in m which is difficult to solve. Thus, we are going to approximate it by an easier to compute closed expression. Since the translation density is modeling the effect of noise, the effective support of $f(s|r)$ is going to be concentrated around s. Then, we can substitute r^{m-1} in (5) by its Taylor expansion around R_i. Let us write (5) in the following way

$$m(x_t) = I^{-1} = \left[\frac{1}{k-1}\sum_{i=1}^{k-1} I_i\right]^{-1}, \quad (6)$$

and expand r^{m-1} in the integral I_i via its Taylor series

$$I_i := \frac{\int_0^{R'} f(R_i|r) r^{m-1} \log \frac{R_k(x_t)}{r} dr}{\int_0^{R'} f(R_i|r) r^{m-1} dr}$$
$$= \frac{\int_0^{R'} f(R_i|r) \log \frac{R_k(x_t)}{r} dr + \Delta I_{N_i} + \dots}{\int_0^{R'} f(R_i|r) dr + \Delta I_{D_i} + \dots} = \frac{I_{N_i}}{I_{D_i}},$$

where

$$\Delta I_{N_i} := (m-1) R_i^{-1} \int_0^{R'} f(R_i|r)(r - R_i) \log \frac{R_k(x_t)}{r} \, dr, \tag{7}$$

and

$$\Delta I_{D_i} := (m-1) R_i^{-1} \int_0^{R'} f(R_i|r)(r - R_i) dr. \tag{8}$$

These integrals are small since the effective support of $f(R_i|r)$ has the same order than the level of noise (considered not very large), and the quantity $(r - R_i)$ is small in the vicinity of R_i. We can then approximate

$$I_i \approx \frac{\int_0^{R'} f(R_i|r) \log \frac{R_k(x_t)}{r} dr}{\int_0^{R'} f(R_i|r) dr}. \tag{9}$$

Notice that with this approximation of I_i, the estimator (6) still reduces to the noise-free Levina-Bickel estimator (2), that is $I_i = \log \frac{R_k}{R_i}$, when $f(R_i|r) = \delta(R_i - r)$. In the more general case, (9) is the expected value of $\log \frac{R_k}{r}$ according

to the transition density $f(R_i|r)$ and thus reducing the effect of noise. Using the approximation (9) in (6) we obtain

$$m(x_t) \approx \left[\frac{1}{k-1} \sum_{i=1}^{k-1} \frac{\int_0^{R'} f(R_i|r) \log \frac{R_k}{r} dr}{\int_0^{R'} f(R_i|r) dr} \right]^{-1}. \quad (10)$$

We explicitly estimate, in the following Section, the error produced in $m(x_t)$ when we use the approximation (10) instead of (5), for the particular important case of a Gaussian transition density,

$$f(s|r) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(s-r)^2}{2\sigma^2} \right). \quad (11)$$

In this particular case that the coordinates are perturbed by Gaussian noise, the error in the Euclidean distance can be approximated by a Gaussian as well (see Appendix A for more details). Thus, the expression for the local dimension estimator becomes

$$m(x_t) \approx \left[\frac{1}{k-1} \sum_{i=1}^{k-1} \frac{\int_0^{R'} \exp\left(-\frac{(R_i-r)^2}{2\sigma^2} \right) \log \frac{R_k}{r} dr}{\int_0^{R'} \exp\left(-\frac{(R_i-r)^2}{2\sigma^2} \right) dr} \right]^{-1}. \quad (12)$$

3.1 Approximation error for a Gaussian translation density

In order to estimate the error of approximating (5) by (10), we compute the integrals (7) and (8), which are the largest order error terms of the numerator and denominator, respectively, in the approximation of $m(x_t)$. For the integral (8), notice that the Gaussian is even with respect to R_i and that $(r - R_i)$ is odd. Then, (8) is zero if the effective support of the Gaussian is within the interval $[0, R']$, that is essentially if $R_i \in [3\sigma, R' - 3\sigma]$. If $R_i \in [0, 3\sigma] \cup [R' - 3\sigma, R']$, (8) is bounded by $4.5\sigma^2(m-1)/R_i$. We will use this bound for ΔI_{D_i} independently of the value of R_i. Regarding the integral (7), we use the Taylor expansion of $(r - R_i)\log\frac{R_k}{r}$ around R_i,

$$(r - R_i)\log\frac{R_k}{r} = (r - R_i)\log\frac{R_k}{R_i} - \frac{(r - R_i)^2}{R_i} + \dots.$$

Again, we consider the worst case scenario, $R_i \in [0, 3\sigma] \cup [R' - 3\sigma, R']$, and we obtain

$$\Delta I_{N_i} \leq 4.5\sigma^2\frac{m-1}{R_i}\log\frac{R_k}{R_i}.$$

We use these bounds and error propagation theory to obtain the relative error on I_i,

$$\frac{\Delta I_i}{I_i} = \frac{\Delta I_{N_i}}{I_{N_i}} + \frac{\Delta I_{D_i}}{I_{D_i}} = 4.5\sigma^2\frac{m-1}{R_i}\left(\frac{1}{I_{N_i}}\log\frac{R_k}{R_i} + \frac{1}{I_{D_i}}\right),$$

and the relative error on $m_k(x_t)$,

$$\frac{\Delta m(x_t)}{m(x_t)} = \frac{\Delta I}{I} = \frac{1}{I(k-1)} \sum_i \Delta I_i,$$

which is bounded by

$$\frac{\Delta m(x_t)}{m(x_t)} \leq \frac{4.5\sigma^2(m(x_t)-1)}{\min_i \left(R_i \tilde{R}_i^{\,m(x_t)-1} \right)} \left(1 + \frac{m(x_t)}{m(x_t, \sigma = 0)} \right), \tag{13}$$

where $m(x_t, \sigma = 0)$ is (2), or equivalently, (5) with $\sigma = 0$, and $\tilde{R}_i^{\,m-1} = I_{D_i} = \int_0^{R'} f(R_i|r) r^{m-1} dr$. This provides a bound on the error of the approximation for the important case of Gaussian noise. Similar computations can be performed for other translation density (noise models). In the case of $\sigma = 0$ (no noise), the approximation error $\Delta m(x_t)$ is zero, as expected. If we consider $\tilde{R}_i \approx R_i$, the bound (13) is inversely proportional to the signal to noise ratio and proportional to $(m-1)/R_i^{m-2}$, which is a decreasing function of the dimension m for $R_i > 1$. Note that the estimator $m(x_t)$, defined in (5), is invariant to distance rescalings so we can always ensure $R_i > 1$.

4 Dimensionality and density estimation with simultaneous soft clustering

Having introduced the critical translational Poisson model, we are now ready to introduce the mixture of these models to address the problem of stratification learning for noisy point cloud data. We start with the basic model, and then introduce a regularization term. We conclude the presentation providing asymptotic results.

4.1 Translation Poisson Mixture Model (TPMM)

In [15], we proposed to study a stratification by extending the Levina and Bickel's technique. Instead of modeling each point and its local ball of radius R as a Poisson process and computing the maximum likelihood (ML) for each ball separately, all the possible balls are considered at the same time in the ML function. The probability density function for the whole point cloud becomes a mixture of Poisson distributions with different parameters (dimension and density) in each class. This allows for the presence of different intrinsic dimensions and densities in the dataset. These are automatically computed while being used for soft clustering. We extend this approach here to the more general case when we have mixtures of translated Poisson processes (thereby handling the noise).

Let us consider J different translated Poisson distributions in the mixture, each one with a (possibly) different dimension m and density parameter θ. Let us denote by ψ the vector set of parameters, $\psi = \{\psi^j = (\pi^j, \theta^j, m^j); j = 1, \ldots, J\}$, where π^j is the mixture coefficient for class j (the proportion of distribution j in the dataset), θ^j is its density parameter ($\rho^j = e^{\theta^j}$), and m^j is its dimension. While in the Levina-Bickel approach the density is assumed locally constant (inside a ball) here the density is assumed constant inside a class (a single Poisson distribution defines each class). However, if there is a class with different densities the algorithm will cluster also according to density (not only dimension) and a single manifold will be represented by clusters of the same dimension but different densities. An example of that is shown in Figure 5. If the number of classes is not sufficient to represent the dimension and density variability, the algorithm will give one or more classes with a dimension and/or density which are the (weighted) average of the actual features within the class. This is the standard result for under-clustering. On the other hand, we have experimentally observed that giving extra classes is reasonable robust, since the extra classes end-up being empty or identical to other classes.

The observable event is, as in the Levina-Bickel approach, the number of points inside the ball $B(R, x_t)$ of radius R centered at point x_t, denoted by $y_t = N(R, x_t)$. The total number of observations is T' and $Y = \{y_t; t = 1, \ldots, T'\}$ is the observation sequence. Often, $T' \equiv T$, all points in the dataset are considered. Let us also denote by $p(\cdot)$ the probability density function and by $P(\cdot)$ the probability. The density function of the Poisson mixture model is given by

$$p(y_t|\psi) = \sum_{j=1}^{J} \pi^j p(y_t|\theta^j, m^j).$$

Since the observations follow a Poisson distribution, and we use the translated Poisson model introduced in the previous section, we have

$$p(y_t|\theta^j, m^j) = e^{\int_0^R \log \mu^j(s)\, dN(s,x_t)} e^{-\int_0^R \lambda^j(r)dr},$$

where $\lambda^j(r) = e^{\theta^j} V(m^j) m^j r^{m^j-1}$ and $\mu^j(s) = \int_0^{R'} f(s|r) e^{\theta^j} V(m^j) m^j r^{m^j-1} dr$. If Y contains T statistically independent variables (a standard assumption), then the probability density function of the observation sequence is the product of the individual probability densities, $p(y_t|\psi)$, and the log-likelihood is

$$L(Y|\psi) = \log p(Y|\psi) = \sum_{t=1}^{T} \log p(y_t|\psi). \qquad (14)$$

Let us consider the hidden-state information, that is, which mixture (or expert) generates each observation. We denote

by $Z = \{z_t \in C; t = 1, \ldots, T\}$ the set of hidden variables and by $C = \{C^1, C^2, \ldots C^J\}$ the set of class labels. Then, $z_t = C^j$ means that the j-th mixture generates y_t. Using Z we can write the complete data log-likelihood as

$$\log p(Z, Y|\psi) = \sum_{t=1}^{T} \sum_{j=1}^{J} \delta_t^j \log \left[p(y_t|\psi^j)\pi^j \right], \qquad (15)$$

where a set of indicator variables δ_t^j, called membership functions, is used in order to indicate the status of the hidden variables:

$$\delta_t^j \equiv \delta(z_t, C^j) = \begin{cases} 1 & \text{if } z_t = C^j, \\ 0 & \text{otherwise.} \end{cases}$$

The unknown parameters in (15) are: The membership function of an expert (class), δ_t^j, the mixture probabilities, π^j, and the parameters of each expert, m^j and θ^j. Usually, problems involving a mixture of experts are solved by the Expectation Maximization (EM) algorithm [10] [21, Chap. 3]. The EM is based on the following decomposition of the log-likelihood (14):

$$\begin{aligned} L(Y|\psi, H) = & \sum_{t=1}^{T} \sum_{j=1}^{J} h^j(y_t) \log \left[p(y_t|\psi^j)\pi^j \right] \\ & - \sum_{t=1}^{T} \sum_{j=1}^{J} h^j(y_t) \log \left[h^j(y_t) \right], \end{aligned} \qquad (16)$$

where $H = \{h^j(y_t) \leq 1; t = 1, \ldots, T, j = 1, \ldots, J\}$ and $h^j(y_t)$ is the probability that observation t belongs to mixture j: $h^j(y_t) = E_Z[\delta_t^j | y_t, \psi] = P(\delta_t^j = 1 | y_t, \psi)$, where $E_Z(\cdot)$ is the expectation with respect to Z. Since the membership functions are indicator variables, the first term in (16) is the expectation of (15) with respect to Z. Also notice that the second term is the entropy of the membership functions.

An interesting interpretation of the EM algorithm is introduced in [17], where the EM is seen as an alternate optimization algorithm of the log-likelihood (16). Then, the E-step is nothing else than the maximization of $L(Y|\psi, H)$ with respect to H with the additional constraint that $\sum_{j=1}^{J} h^j(y_t) = 1$ for each observation $t = 1, \ldots, T$. Thus, the variables $h^j(y_t)$ at step $n+1$ of the optimization algorithm are

$$h_{n+1}^j(y_t) = \frac{p(y_t|m_n^j, \theta_n^j)\pi_n^j}{\sum_{l=1}^{J} p(y_t|m_n^l, \theta_n^l)\pi_n^l}. \tag{17}$$

In the same way, variables ψ are obtained by maximizing $L(Y|\psi, H)$ with respect to ψ with an additional constraint for the mixture probabilities: $\sum_{j=1}^{J} \pi^j = 1$. This gives equations (21)-(23) for the variables at step $n+1$. In order

to compute m_{n+1}^{j} we have used the same approach as in [23], by means of a k nearest neighbor graph. The TPMM approach just described is summarized in **R-TPMM Algorithm** below, for the particular case of $\alpha = 0$ (no regularization, see below).

4.2 Regularized TPMM

The TPMM algorithm seeks a soft clustering according to dimensionality and density, considering noise in the data, but does not (explicitly) take into account spatial information. Adding regularization is the goal of this section. Regularization further helps to improve the classification in noisy data and points lying close to manifold edges (see results in figures 1 and 2). This regularization is inspired in part by the work in [1] for the neighborhood EM (NEM), where the authors extend the EM algorithm adding spatial constraints. This neighborhood spatial information is introduced as a penalization term in the log-likelihood, following Hathaway's EM interpretation [17]. In our context, we complete (16) with a spatial term $S(H)$,

$$F(\psi, H) = L(Y|\psi, H) + \alpha S(H), \tag{18}$$

where α is a parameter that controls the tradeoff between the spatial term and the likelihood. Its value is also related to the amount of noise in the data.[2] Then, function F is maximized with an alternate optimization technique. Since the new term, S, only depends on H, the optimization procedure results in a EM-type algorithm with a modified membership probability that not only depends on the likelihood but also on the spatial criteria. The NEM algorithm uses (note the similitude with MRFs, see below)

$$S_{NEM}(H) = \sum_{t=1}^{T} \sum_{j=1}^{J} h^j(y_t) \sum_{l \sim t} h^j(y_l),$$

where $l \sim t$ indicates that there is a neighborhood relationship between observations l and t. By maximizing this term, we want, for each observation t, as many neighbors as possible with high probability of belonging to the same class as observation t, thus regularizing the classification. However, we will use a more general expression for $S(H)$ based on a dissimilarity measure, $\mathcal{D}$, between every observation and other observations in the sequence,

$$S(H) = -\sum_{t=1}^{T} \sum_{j=1}^{J} h^j(y_t) \mathcal{D}(t, j, X, H). \qquad (19)$$

The expression (19) provides a generic framework for introducing constraints in the soft classification, besides the ones already present in the TPMM model, namely dimensionality and density. One possibility, as in the NEM algorithm, is to introduce spatial regularity. Then, as dissimilarity measure we use $\mathcal{D} = \mathcal{D}_R$ defined as

$$\mathcal{D}_R := \sum_{l \sim t} (1 - h^j(y_l)).$$

Different neighborhoods definitions in $\mathcal{D}_R$ result in different kinds of regularization. A natural choice is the manifold neighborhood, for that, we can define as neighbors the k nearest neighbors. However, for specific applications one might be interested in other neighborhoods, e.g., pixel neighborhoods or contiguous frames in video applications (see experiment in Figure 10 and Table 5).

As noted in [1], the EM algorithm with additional constraints can be seen as finding the Gibbs distribution with energy $-F(\psi, H)$. In the particular case when the additional constraint is neighborhood dependent, $S_{NEM}(H)$ and $S(H)$ with $\mathcal{D}_R$, the Gibbs distribution defines a Markov Random Field.

The maximization of F (Equation (18)), is obtained as in [1], with an alternate optimization technique which results in an EM-type algorithm. Maximizing (18) with respect to H, with $S(H)$ defined in (19) – with the constraints $\sum_{j=1}^{J} h^j(y_t) = 1$ for each observation $t = 1, \ldots, T$, by means of Lagrange multipliers – results in the following expression for the membership probabilities:

$$h^j(y_t) = \frac{p(y_t|m^j, \theta^j)\pi_n^j e^{-\alpha \mathcal{D}'(t,j,X,H)}}{\sum_{l=1}^{J} p(y_t|m^l, \theta^l)\pi^l e^{-\alpha \mathcal{D}'(t,l,X,H)}}, \qquad (20)$$

where $\mathcal{D}'(t, l, X, H) = \sum_{l \sim t}(1 - 2h^j(y_l))$ in the particular case we are interested: $\mathcal{D} = \mathcal{D}_R$, and assuming that $l \sim t$ implies $t \sim l$. Since the only term in (18) which depends on ψ is $L(Y|\psi, H)$, the optimal values of $\psi^j = \{(\pi^j, \theta^j, m^j)$ for $j = \{1, \ldots, J\}\}$ do not change with respect to the original TPMM algorithm. The regularized version of the TPMM algorithm is summarized in the **R-TPMM Algorithm** below (Regularized Translated Poisson Mixture Model Algorithm).

The EM suffers from local maxima, this can be alleviated running the algorithm several times with different initializations. In particular, we add to the EM iterations an extra loop where the parameters m^j and θ^j of each class are reinitialized every odd iteration and π^j every even iteration.

level or full noise/translation function f).
ENSURE: Regularized soft clustering according to dimensionality and density.

1. Compute the local estimators

$$m(x_t) = \left[\frac{1}{k-1} \sum_{j=1}^{k-1} \frac{\int_0^{R'} f(R_i(x_t)|r) \log \frac{R_k(x_t)}{r} dr}{\int_0^{R'} f(R_i(x_t)|r) dr} \right]^{-1}$$

$$\theta(x_t) = \log \left((k-1) / \left(V(m(x_t)) R_k(x_t)^{m(x_t)} \right) \right)$$

   In particular, we use the definition of f given in (11).

2. Initialize $\psi_0 = \{\pi_0^j, m_0^j, \theta_0^j\}$ and $\bar{\psi}_0 = \{\bar{\pi}_0^j, \bar{m}_0^j, \bar{\theta}_0^j\}$ to any set of values which ensures that $\sum_j \pi_0^j = \sum_j \bar{\pi}_0^j = 1$ and $\bar{H}_0 = \{\bar{h}_0^j(y_t) = 1/J;\ j = 1, \ldots, J,\ t = 1, \ldots, T\}$.

3. Iterations on l,

   3A. If l is odd

   Set $\bar{m}_l^j = m_0^j$ and $\bar{\theta}_l^j = \theta_0^j$, for all $j = 1, \ldots, J$.

   Else

   Set $\bar{\pi}_l^j = 1/J$, for all $j = 1, \ldots, J$.

   3B. Iterations on n,

   For all $j = 1, \ldots, J$:

   3B.1: Compute, for all $t = 1, \ldots, T$,

$$h_{n+1}^j(y_t) = \frac{p(y_t|m_n^j, \theta_n^j) \pi_n^j e^{-\alpha \mathcal{D}'(t,j,X,H_n)}}{\sum_{l=1}^J p(y_t|m_n^l, \theta_n^l) \pi_n^l e^{-\alpha \mathcal{D}'(t,l,X,H_n)}},$$

   where $H_n = \{h_n^j(y_t);\ j = 1, \ldots, J,\ t = 1, \ldots, T\}$.

   3B.2: Compute

$$\pi_{n+1}^j = \frac{1}{T} \sum^T h_n^j(y_t) \tag{21}$$

$$\pi^j_{n+1} = \frac{1}{T}\sum_{t=1}^{T} h^j_n(y_t) \tag{21}$$

$$m^j_{n+1} = \left[\sum_{t=1}^{T} h^j_n(y_t) m(x_t)^{-1} / \sum_{t=1}^{T} h^j_n(y_t)\right]^{-1} \tag{22}$$

$$\rho^j_{n+1} = e^{\theta^j_{n+1}} = \left[\sum_{t=1}^{T} h^j_n(y_t) f(x_t)^{-1} / \sum_{t=1}^{T} h^j_n(y_t)\right]^{-1} \tag{23}$$

where $\rho(x_t) = e^{\theta(x_t)}$.

Until convergence of ψ_n, that is, when $||\psi_{n+1} - \psi_n||_2 < \epsilon$, for a certain small value ϵ.

Set $\bar{\psi}_{l+1} = \psi_n$ and $\bar{H}_{l+1} = H_n$.

Until $||\bar{\psi}_{l+1} - \bar{\psi}_l||_2 < \epsilon$, $||\bar{H}_{l+1} - \bar{H}_l||_2 < \epsilon$ or $l = l_{\max}$.[3]

Remark 1. *The PMM and R-PMM algorithms introduced respectively in [15] and [16] are particular cases of the parameters α (regularization) and σ (noise) in the R-TPMM algorithm . Let us introduce the following notation for the particular cases of these parameters:*

- *PMM:* $\alpha = 0$ *and* $\sigma = 0$.
- *R-PMM:* $\alpha > 0$ *and* $\sigma = 0$.
- *TPMM:*$\alpha = 0$ *and* $\sigma > 0$.
- *R-TPMM:*$\alpha > 0$ *and* $\sigma > 0$.

We will use the above notation in the experiments in Section 5.

Remark 2. *Notice that the estimators* (22)-(23) *in the PMM and R-PMM approaches ($\sigma = 0$) are weighted harmonic means of the local estimators* (2)-(3) *of Levina-Bickel. The weight at each point is the probability of the membership function, h. In the particular case of a unique class, $J = 1$, we obtain the global dimension estimator proposed by MacKay and Ghahramani (http://www.inference.phy.cam.ac.uk/mackay/dimension/), a particular case of our proposed framework.*

Remark 3. *We are using the same level of noise σ for all the clusters. A better approach might be to use a σ suitable for each class. Although computationally speaking it will be more demanding, since we would have to recompute the local estimators $m(x_t)$ and $\theta(x_t)$ at each iteration with the σ of the assigned class. Moreover, the different σ would have to be estimated (this can be done for example as the value of σ which minimizes the estimated dimension in each class).*

As proved in [2], if α is small enough, (18) has a guaranteed global maximum for a fixed value of ψ, and the additional term $S(H)$ does not affect the convergence of the EM-type algorithm. It can be shown (see Appendix B) that, for the case of $\mathcal{D}_R$, the corresponding bound on α is $\alpha_R < 1/(2k)$.

Using the same analysis as in Section 3.1 we find that the relative error produced in (22) by using the approximation (10) for $m(x_t)$ is

$$\frac{\Delta m^j}{m^j} \leq \frac{4.5\sigma^2(m^j - 1)}{\min_{i,t}\left(R_i(y_t)\tilde{R}_i(y_t)^{m^j-1}\right)}\left(1 + \frac{m^j}{m^j(\sigma = 0)}\right),$$

where $m^j(\sigma = 0)$ is (22) with $\sigma = 0$, and $\tilde{R}_i(y_t)^{m^j-1} = I_{D_i}(y_t)$.

4.3 Asymptotic analysis

Levina and Bickel show in [23] that under the assumptions $T \to \infty$, $k \to \infty$, and $k/T \to 0$, that is when the Poisson approximation is correct, the mean and variance of the dimension estimator (2) (with $k-2$ instead of $k-1$ in the denominator) are

$$E[m(x_t)] = \mathrm{m}_T, \quad \mathrm{Var}[m(x_t)] = \frac{\mathrm{m}_T^2}{k-3},$$

where m_T is the actual dimension. We can apply the same type of analysis to our model in the particular case of hard clustering, that is

$$h^j(y_t) = \begin{cases} 1 & \text{if } j = \mathrm{argmax}_i h^i(y_t), \\ 0 & \text{otherwise.} \end{cases}$$

We assume, in addition, that all the points that belong to class j are well classified. Then, we obtain the following results

$$E[m^j] = \mathrm{m}_T^j + \frac{\mathrm{m}_T^j}{(k-1)N_j - 1},$$

$$\mathrm{Var}[m^j] = (\mathrm{m}_T^j)^2 O\left(\frac{1}{(k-1)N_j - 4}\right),$$

where m_T^j is the correct intrinsic dimension of class j and N_j is the amount of points classified as class j. See Appendix C for the details of the proof. This result shows that the dimension estimator of each class is more biased when the intrinsic dimension increases. On the other hand, when there are more points in a class (N_j is larger), the bias is reduced. It is reduced also by considering more nearest neighbors, although there is a compromise for this value since increasing k affects the underlying hypothesis of constant density inside the ball. We have verified this result experimentally and we found that the bias results in a underestimation of the intrinsic dimension (this behavior was also observed in the Levina-Bickel estimator [23]), and that it depends on the intrinsic dimension but not on the ambient dimension. We have also experimentally observed that a possible bias in the estimated dimension does not affect the clustering, unless the bias makes the estimated dimension of one class to be close to any of the other clusters estimated dimensions.

The analysis of the density estimator θ^j is the subject of current research, as it is the study of the asymptotic behavior for the full soft clustering model.