GMDA: Multivariate two-sample tests

Antoine Guillot

June 11, 2018

1 ToMATo

The ToMATo algorithm was downloaded from INIRA and compiled. I built a quick Python wrapper to be able to call the C++ algorithm from Python.

1.1 Parameters selection

The original paper proposes a way to select the merging parameter τ. They first represent the topological peaks on a persistence diagram, the x-axis represents the birth level of a peak and the y-axis represents the level at which a peak is merged with a higher peak. The persistence of a point is $p_i = \frac{|level_{birth} - level_{death}|}{\sqrt{2}}$, by sorting the points by their persistence, we can compute the persistence gaps between consecutive points. When doing so, we need to remove the highest peak, since it never dies its persistence is $+\infty$, this would bias the selection of τ (Since τ would always be between the highest peak and the second highest one).
The paper advises to select τ, such that it lies belong the two points that maximize the persistence gaps. Hence, I implemented the algorithm in order to:

1. Compute the persistence diagram, by setting $\tau = +\infty$
2. Compute the maximum persistence gap between two points x and y such that p_x and p_y are finite. Set $\tau = \frac{(p_x + p_y)}{2}$
3. Compute ToMATo again, with the optimal computed τ

However, the algorithm also require to select two other hyper parameters, the number of neighbors k used to compute the distance to the measure and the parameter δ of the Rips graph.
This parameter δ selects the distance with which two points should be connected. Low values of δ would focus on local structure and possibly noise, on the other hand high values of δ will make the algorithm look at the data on a larger scale. I found this parameter to be critical to have a good clustering, hence an automated way to compute an optimal δ.

1.2 Automatic selection of δ

1.2.1 Overview

First we need to find an optimality criterion for δ and k.We ordered the points of the samples by their persistence and only keeps the points with a finite persistence: $E = \{x \in \mathbb{X} \; |p_x| < +\infty\}$ We denote $P* = max_{x,y \in \mathbb{X}} |p_x - p_y|$, the maximum persistence gap between two points. We can also see $P*$ as a measure of the quality of the clustering, maximizing $P*$ would advocate for a good clustering. Hence, the optimal value of δ is the value which will maximize the maximum persistence gap.

1.2.2 Algorithm (Grid-Search)

We denotes $x_0, .., x_n$
Set $max_{PD} = 0$.
Set $opt_{delta} = 0$.
for x in $x_0, .., x_n$:

1. compute the optimal τ when $delta = x$ and save the associated maximum PD gap $PD_{candidate}$.
2. If $PD_{candidate} > max_{PD}$,set $max_{PD} = PD_{candidate}$ and set $opt_{delta} = x$

Return opt_{delta}
The algorithm can easily be parallelized and yielded better results than more advanced optimization methods (probably because the PD gap does not evolves smoothly with $delta$).

1.2.3 Results

We used 3 datasets:

- Gaussian blobs with 25 blobs dimension from 2 to 20 and 12500 points.
- 2 Moons generated with scikit-learn with uniform noise.

On these datasets, selecting δ which maximize the persistence gap would very often compute the right number of clusters and yield a good clustering when the space is low-dimensional. The numbers of neighbors k is arbitrarily set to 50.

As we can see on the different pictures, maximizing the PD gap with respect to δ yields good and automatic clustering on the several datasets. The PD gap sensibly increases when it reach the "natural" scale of the data.(As a sidenote: If the dataset had several "natural" scales would we see several peak and plateau ?)
As the dimensionality of the dataset increases, the difficulty of the task also increase, the interval of δ for which the number of clusters is correct is narrower. Since the PD gap curve is not convex, so we will use a grid search algorithm to find the optimal parameters in the next section.

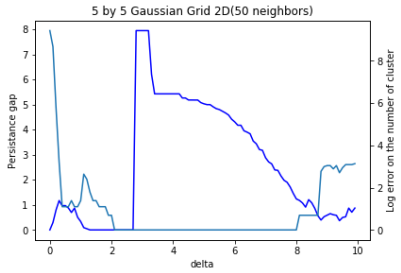


Figure 1: Evolution of the PD gap on 2D Gaussian blobs

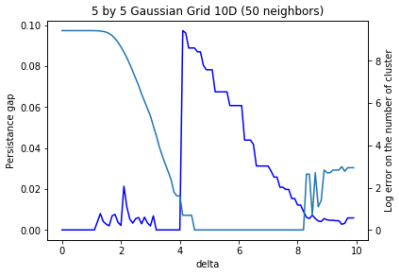


Figure 2: Evolution of the PD gap on 10D Gaussian blobs

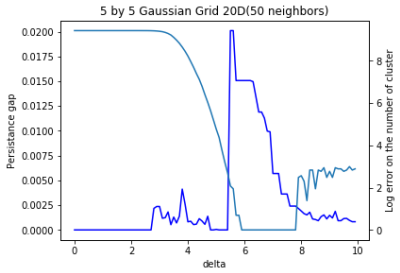


Figure 3: Evolution of the PD gap on 20D Gaussian blobs

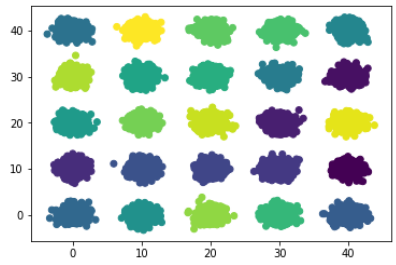


Figure 4: Clusters in the 2D Gaussian blobs

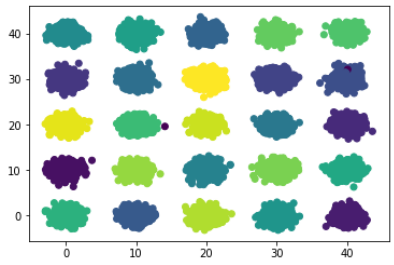


Figure 5: Clusters in the 10D Gaussian blobs

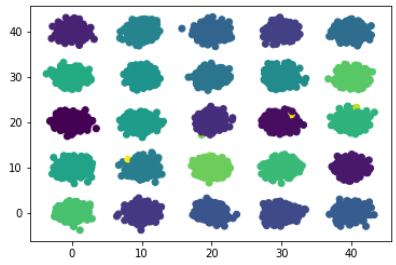


Figure 6: Clusters in the 20D Gaussian blobs

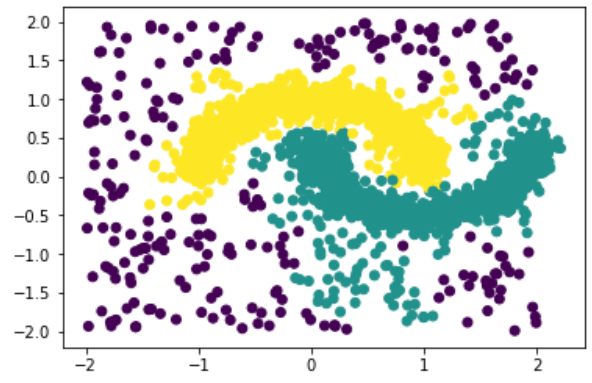


Figure 7: Detected clusters in the 2 Moons dataset

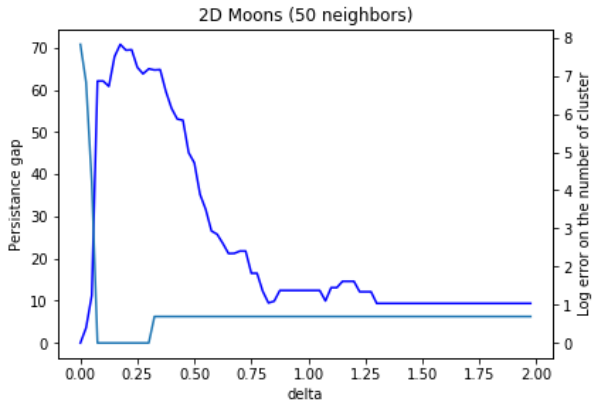


Figure 8: Evolution of the PD gap

The current criterion could probably be improved by smoothing the PD value or by taking into account the norm derivative. As we can see on Figure 1 to 3, that the clustering is stable around the optimal *delta* and so should be the PD gap.

1.2.4 Limits

The algorithm perform well on low dimension data. However, in large dimensional space, the interval where the PD gap is close to the optimal delta is getting narrower. This will

be a real issue when dealing with large dimensions, we won't be able to correctly assess the 'natural' scale of the data and the estimation of the kernel width will be wrong.

2 MMD two-sample test and kernel width selection

In this section we implemented a bootstrapped maximum mean discrepancy test, the width of the Gaussian kernel was selected using the ToMATo algorithm.

2.1 Bootstrapped MMD test

2.1.1 MMD^2 estimate

The maximum mean discrepancy between two probability distribution P and Q over a functional space $\mathcal{F}$ is:

$$MMD[F, P, Q] = sup_{f \in F} | \mathbb{E}_{X \sim P}[f(X)] - \mathbb{E}_{Y \sim Q}[f(Q)] |$$

If $\mathcal{F}$ is the unit ball in an RKHS, the MMD can be rewritten using the r.k of the RKHS:

$$MMD^2[F, P, Q] = \mathbb{E}_{X,X'}[k(X, X')] + \mathbb{E}_{Y,Y'}[k(Y, Y')] - 2\mathbb{E}_{Y,X}[k(Y, X)]$$

X, X' are iid and follow P while Y, Y' are also iid and follow Q. From this theoretical formula, we can derive the unbiased empiric estimate of the MMD^2 of two samples $x^{(n)}$ and $y^{(m)}$ of respective size n and m:

$$MMD_u^2[F, x^{(n)}, y^{(m)}] = \frac{\mathbf{G}_k(x,x) - Tr(\mathbf{G}_k(x,x))}{n.(n-1)} + \frac{\mathbf{G}_k(y,y) - Tr(\mathbf{G}_k(y,y))}{m.(m-1)} - \frac{2.\mathbf{G}_k(x,y)}{m.n}$$

In this equation, $\mathbf{G}_k$ denotes the Gram matrix associated with the kernel k.We removed the indices to make the equation lighter.

2.1.2 Bootstrapped test

Direct inferences using the MMD^2 estimates and the acceptance threshold derived in the class (Corr.11) yielded bad results. Due to this, we used Bootstrapped MMD^2 test. The procedure is the following:

1. Compute $MMD_u^2[F, x^{(n)}, y^{(m)}]$
2. Create a merged sample of $x^{(n)}$ and $y^{(m)}$ denoted $z^{(m+n)}$
3. Draw (with replacement) two samples from $z^{(m+n)}$ and compute their MMD^2 estimate. Repeat N times and store the value to compute the empirical distribution Q of the MMD^2 under the null hypothesis.
4. if $MMD_u^2[F, x^{(n)}, y^{(m)}] > Q_{1-\alpha}^{-1}$ reject the null hypothesis. Otherwise the test in inconclusive.

The underlying idea is that if $x^{(n)}$ and $y^{(m)}$ are from the same distribution (null hypothesis) sampling from the merged sample should yield a similar MMD^2 (which will be close to 0).

2.2 Kernel and kernel width computation

We used a RBF kernel:

$$k(x, y) = exp(-\frac{||x-y||^2}{2.\sigma^2})$$

The kernel width σ is estimated by using the median distance between point in the same clusters. The clustering is computed using the ToMATo algorithm with the choice of optimal parameters τ and δ as described in the Question 1. Since τ is automatically computed we did not try several thresholds.

2.3 Overview

In this section, we propose a novel two sample test based on the Wilcoxon Mann Whithney test. Let $x^{(n)}$ and $y^{(m)}$ be two iid sample respectively from the distribution P and Q which are defined over $\mathbb{R}^N$. We denote u a random unitary vector from $\mathbb{R}^N$. Then $\forall x \in x^{(n)}$ and $\forall y \in y^{(m)}$ we can define $x' = x^T u$ and $y' = y^T u$, the projection of the points along u.
$y'^{(n)}$ and $x'^{(m)}$ are now random variables in $\mathbb{R}$, on which the Wilcoxon Mann Whithney test can be ran. However, we are not running the test on the initial populations but on their projections.
Hence the original null hypothesis: $H_0 : P = Q$ becomes $H'_0 : P' = Q'$ where P' (resp Q') is the theoretical distribution of a random variable following P (resp Q) and projected on u.

2.4 Implementation and results

The multivariate U-test yield very poor results independently of the dimensions or the difficulty of the dataset. It was not able to discriminate the dataset and the type-II error stayed around 99.5%

3 Kernel PCA U-test

Due to the very poor results of our implementation of the multivariate U-test, we propose a novel methods for multivariate two samples test.

3.1 Overview

Let $x^{(n)}$ and $y^{(n)}$ be two samples drawn from two distributions P and Q defined on $\mathbb{R}^N$. Let $\mathcal{F}$ be a functional space which is the unit in a RKHS, we denote k the reproducing kernel of this RKHS. $\mathbf{G}_k$ denotes the Gram matrix associated with the r.k. k, then, $G_k(x^{(n)}, x^{(n)})$ and $G_k(y^{(n)}, y^{(n)})$ are two symmetric matrices and can be diagonalized, we denote $\lambda_1, ..\lambda_n$ and $\mu_1, .., \mu_m$ their respective eigenvalues. We can now define $\mathbf{L}_{\lambda n}$ and $\mathbf{L}_{\mu_m}$ the spectral measure associated with the two Gram matrices. Asymptotically, under the null hypothesis, L_λ and L_μ should have the same distribution.
Hence we can now associate to each sample the sequence of its eigenvalues, and compare this sequence of real number with a U-test instead of having to compare the two samples

in $\mathbb{R}^N$.With a proper kernel selection, this technique yields good results for a reasonable computational cost.

3.2 kPCA-TST algorithm

Our implementation uses a Gaussian Kernel: $k(x, y) = exp(-\frac{||x-y||^2}{2.\sigma^2})$, where σ is the Kernel width. α is our significance threshold.

1. If required, Compute the optimal kernel width using the algorithm from 1.2.
2. Compute $G_k(x^{(n)}, x^{(n)})$ and $G_k(y^{(n)}, y^{(n)})$
3. Center the two Grams matrices: $\mathbf{G_{centered}} = \mathbf{G} - \frac{1}{n}\mathbf{1_n G} - \frac{1}{n}\mathbf{G 1_n} + \frac{1}{n}\mathbf{1_n G 1_n}$. Where $\mathbf{1_n}$ is the matrix filled with ones.
4. Compute the eigenvalues of the two centered Grams matrices and perform the U-test on the two samples. We denote p_u its p-value. If $p_u < \alpha$ reject the null hypothesis, otherwise the test is inconclusive.

3.3 Limits and improvement

The main limitations of the proposed algorithm are computational:

- Computing the parameter δ requires many iterations of ToMATo. Furthermore, the interval for which δ yields close to optimal clustering decreases while the dimension increase. This require to adapt the grid size accordingly and increase the overall computational complexity.
- The computation of the Gram matrix and the eigenvalues computation are the most prominent memory and computational bottleneck. The Gram matrix requires $O(n^2)$ memory and is $O(n^2.p)$ computational-wise. The diagonalization require $O(n^3)$ computation (or $O(n^2.k)$ if we restrict ourselves to the k greatest eigenvalues).

We will take care of the complexity of the diagonalization by proposing a algorithm which is asymptotically linear, have similar performance to kPCA-TST and can be parallelized and run on-line. The algorithm is detailed in the next section.
Other limitations appears in large dimension. Our method to optimize theδ parameter is not suitable for more than 15-20 dimensions, this could be corrected by improving the optimization method or by projecting the initial data in a lower dimension space (for instance using an auto-encoder).

4 Linear-kPCA-TST algorithm

As exposed in the previous section, the kPCA-TST algorithm is computationally expensive. The Linear-kPCA-TST algorithm solves this complexity issue (given that the optimal kernel width is known) and is asymptotically linear.

4.1 Overview

Instead of working on the whole Gram Matrix. One could work on several smaller Gram matrices $\mathbf{G}_1$, $\mathbf{G}_2$, ...,$\mathbf{G}_N$, where each of these matrices is build using a sample of the data. Each Gram matrix has its spectrum Sp_1, .. ,Sp_N, we denote the union of these spectrum $Sp_p^N = U_{i \leq N}$ where p is the distribution of the observations. According to BENAYCH-GEORGES and COUILLET (2016), the empiric distribution of the eigenvalues of large Gram matrix converges to a measure of probability depending only on the distribution of the observations. Hence, we have $lim_{N \leftarrow \inf} Sp_p^N = D_p$ where D_p is a probability distribution which only depends on the original probability distribution p of the observations.
Let $x^{(m)}$ and $y^(m)$ be two samples respectively drawn from two distribution p and q. Then testing the null hypothesis $p = q$ can be seen as testing $D_p = D_q$. To test $D_p = D_q$, we can use a U-test on the observed spectrum (since the eigenvalues are in $\mathbb{R}$).

4.2 Algorithm

Let $x^{(n_1)}$, $y^{(n_2)}$ bet two samples from the distributions to test p and q
Let $\mathbb{S}_{\text{I}}$ and $\mathbb{S}_{\text{II}}$ be two empty lists.

1. If required, Compute the optimal kernel width using the algorithm from 1.2.

2. for i $\leq$ K

   (a) Draw two samples of size n $x^{(n)}$, $y^{(n)}$

   (b) Compute $G_k(x^{(n)}, x^{(n)})$ and $G_k(y^{(n)}, y^{(n)})$

   (c) Center the two Grams matrices: $\mathbf{G_{centered}} = \mathbf{G} - \frac{1}{n}\mathbf{1_n G} - \frac{1}{n}\mathbf{G 1_n} + \frac{1}{n}\mathbf{1_n G 1_n}$. Where $\mathbf{1_n}$ is the matrix filled with ones.

   (d) Compute the eigenvalues of the two centered Grams matrices and append their respective eigenvalues in $\mathbb{S}_{\text{I}}$ and $\mathbb{S}_{\text{II}}$

3. Perform the U-test on $\mathbb{S}_{\text{I}}$ and $\mathbb{S}_{\text{II}}$. We denote p_u its p-value. If $p_u < \alpha$ reject the null hypothesis, otherwise the test is inconclusive.

4.3 Strength and limits

This new algorithm is significantly faster than the original ones and perform on-par on smaller datasets. Its performance on large datasets seems to outperform or be on par with SOTA method. Furthermore, it can be parallelized since the computation of each of the Gram matrix is independent.
However, as kPCA-TST, the Linear-kPCA-TST is also limited to low dimensional spaces when the optimal kernel size needs to be computed.

5 Results

In this section, we present the results given by the three techniques described above, Bootstrapped MMD2,kPCA-TST and linear kPCA-TST.

5.1 Single Gaussian

For comparison sake dataset is similar to the one presented in "A Kernel Two-Sample Test" by Gretton et al. Two Gaussian distributions with respectively unit variance and isotropic variance logarithmically stretched from $10^0.01$ to 10. The two samples contains 250 points and the acceptance threshold is $\alpha = 0.05$. The kernel width is the median distance between points (since there is only one cluster).
The first graph average the type II error over all the variance, while the second one shows a more detailed picture of how variance and dimension plays along in the type II error. This worth noticing that Type I error increases with dimension to such extent that the tests are not relevant for dimensions greater than 500.
Linear kPCA-TST was computed by using 125 observation for the computation of the Gram matrix with 10 repetitions (i.e. the eigenvalue of 10 Gram Matrix were used for both samples). Overall,linear kPCA-TST performs better in low dimensions and similarly to boot-

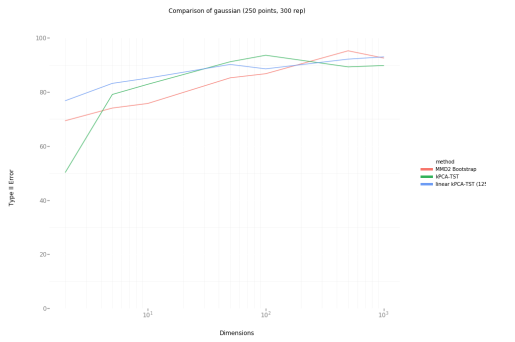


Figure 9: Averaged % of correctly rejected H0 (Gaussian distributions)

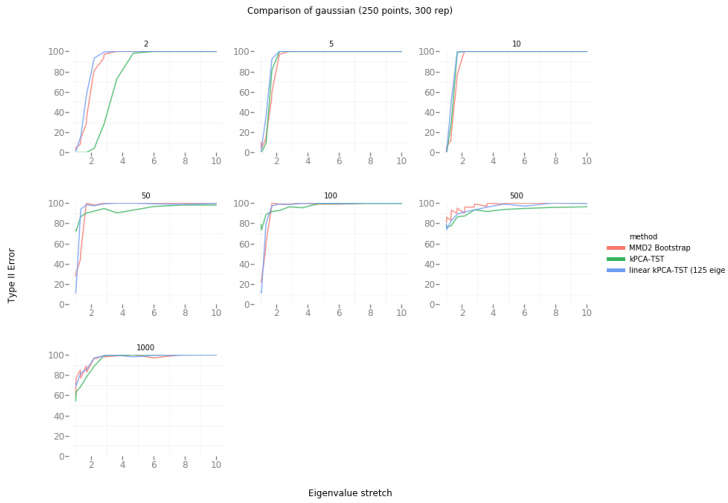


Figure 10: % of correctly rejected H0 (Gaussian distributions)

strapped MMD2 for a lower computational cost. kPCA-TST performs similarly to the two other methods in low dimension but type I error increase faster with dimensions that fro linear kPCA-TST and MMD2.

5.2 Gaussian Blobs 2D

In this section, we replicated the experiments from "Optimal kernel choice for large-scale two-sample tests" from Gretton et al. The grid is 3 by 3 with 2500 points, and 300 repetitions

were done. The kernel choice is done by finding the right number of clusters and then computing the median within cluster distance.

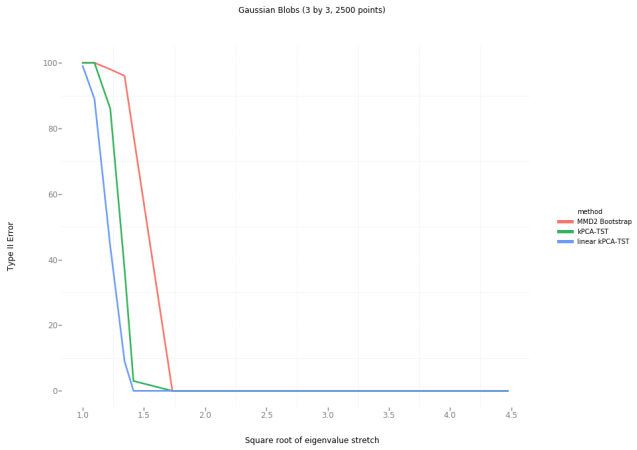
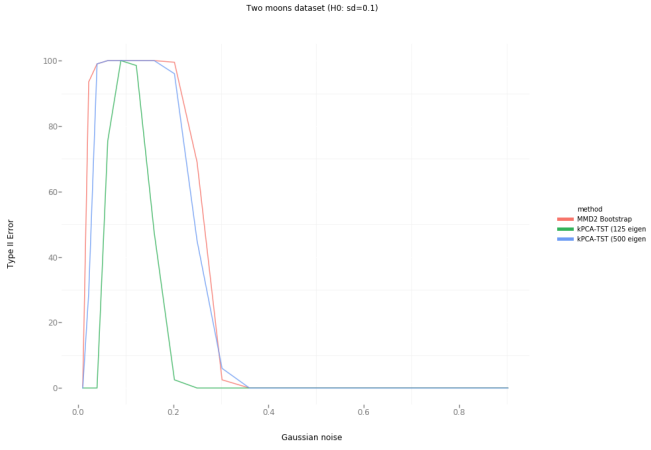


Figure 11: Evolution of Type II errors on Gaussian blobs

The three methods significantly outperform the original papers, which proves our kernel width computation is efficient. Furthermore, kPCA methods seems to outperform bootstrapped MMD2 by a slight margin while being far less computationally intensive.

5.3 Two moons

The last test was ran on the two moons dataset (Figure 7), the first sample was drawn with a Gaussian noise of 0.1 (similar to the on in Figure 7) while the other sample had a varying variance. For each variance, 200 repetitions were ran. The right kernel width was detected with ToMATo and the δ-optimization for all the methods.



Evolution of Type II errors on the moon dataset

Linear kPCA-TST performs better than bootstrap MMD2, the discrimination gap is narrower and the method is able to discriminate the distributions more efficiently.