OOTOC

Release V.0.1

S. Ivvan Valdez

Feb 13, 2024

CONTENTS:

CHAPTER

ONE

TOTAL-OPERATING-CHARACTERISTIC CURVE'S IMPLEMENTATION

This project consist of classes and methods to compute and plot a TOC curve. The operations include, normalization, approximation by vectors (means of subsets of data).

1.1 TOC class

class ootoc.**TOC**(*rank=array([], dtype=float64)*, *groundtruth=array([], dtype=float64)*)

This class implements the Total Operating Characteristic Curve. The TOC computed from a rankl and groundtruth using this instantiation is named **standard TOC** trought the document.

Parameters

- **rank** – The class is instantiated with the optional parameter `rank` that is a numpy array of a predicting feature.
- **groundtruth** – The class is instantiated with the optional parameter `groundtruth` that is a numpy array of binary labels (0,1).

Variables

kind – A string with the value 'None' by default, it indicates the kind of TOC, for instance: TOC, normalized, vector, density and qspline. The `kind` attibute is mainly used for plotting inside the class.

Returns

The class instance, if `rank` and `grountruth` are given it computes the TOC, and the `kind` is TOC, otherwise it is an empty class. Some methods of the TOC class return the other kinds of TOC, suich as density, noprmalize, etc.

Return type

`TOC`

1.1.1 Area

`TOC.`**area = 0**

Area under the curve of the TOC.

1.1.2 Area Ratio

`TOC.areaRatio = 0`

areaRatio with respect to the parallelepiped. If the maximum TOC area is that of the parallelepiped, hence this is the ratio of the TOC area inside the parallelepiped divided by the parallelepiped arena. Notice that the area inside the parallelepiped is usually different from the area under the curve, it is computed substracting from the `area` the triangle in the left side of the parallelepiped.

1.1.3 Base number of positives

`TOC.basenpos = 0`

basenpos preserve the npos value from the standard TOC when it is normalized. Nevertheless it is modified by resample.

1.1.4 Base number of true positve plus false positives

`TOC.basentppfp = 0`

Some TOCs result from applying operations to an standard TOC, in such a case the ntppfp could change int the resulting TOC nevertheless the basentppfp is copied from the standard TOC. For instance, ntppfp is 1 in the normalized TOC, but the basetppfp preserves the value from the standard TOC.

1.1.5 Kind of TOC curve

`TOC.kind = 'None'`

The `kind` attribute indicates the type of TOC curve, that is to say, it is **"TOC"** kind if the x axes presents the true positives plus false positives count, and the y axes presents the true-positive count. It is a **"vector""** kind if the curve is a representation of the original TOC, that is to say it stores *nvector* vectorial means of the original TOC data.

1.1.6 Indices of the sorted rank

`TOC.indices = None`

This attibute stores the indices fo the sorted rank. Storing the indices is useful to save computational time when converting a rank array to a probability value.

1.1.7 Number of data

`TOC.ndata = 0`

This attribute stores the number of data in the arrays of the class, notice that the number of positives (`npos`) or other counts are altered when interpolations, normalization or vectorization of the TOC is applied, while `ndata` stores the length of the data arrays independent of the mentioned counts.

1.1.8 Number of positive instances

TOC.**npos = 0**

Similar to ntpfp, this variable stores the number of positive data (1 valued data), and also it is lost in the normalized TOC (it is 1 in th enormalized TOC).

1.1.9 Number of true positives plus false positives

TOC.**ntppfp = 0**

This is the count fo true positives plus false positives. Notice that this number could be different from the number of data, for instance, normalized, vector, resampled or density TOCs could have a different ndata and ntppfp value.

1.1.10 Positive-class data proportion

TOC.**PDataProp = 0**

Proportion of positive data in the data. The purpose is to maintain the proportion of class 1 data, hence to preserve the knowlkedege about data imbalance and proportion of calsses even if the TOC is normalized.

1.1.11 Thresholds used for the TOC computation

TOC.**thresholds = None**

A numpy array with thresholds of the TOC, they are computed using the ranks, that is to say, in the standard TOC most of the ranks are equal to the thresholds.

1.1.12 Array of true positives plus false postives

TOC.**TPplusFP = None**

A numpy array with the sum of true positives plus false positives.

1.1.13 Array of true positives

TOC.**TP = None**

A numpy array with the sum of true positives

1.1.14 Probability density function computation

TOC.**density**(*smoothing=-1*, *order=1*, *verbose=0*)

This function computes an approximation of the derivative of the TOC curve using cubic and fifth order centered finite differences. That is to say, if the `order` parameter is set to 1, it computes an approximation using a the finite difference before and after the current point, hence this approximation provides a cubic error term. Additionally, the function computes a smooth version of the derivative using a mean filter. The original TP and TPplusFP arrays from the input TOC (the TOC which this function is called from) are copied. .. warning:: It is strongly suggested to use a **vector** or **nvector** kind TOC, because the derivative of a standard TOC only produces finite differences with 0 or 1, hence the derivative of a standard TOC is, usually, highly noisy and non-informative. It is strongly suggested to use a **vector** kind TOC produced from a normalized TOC, or a **nvector** TOC, because this kind of TOC have a maximum value of 1 in both access, hence, the derivative can be seen as a density probability function. This is intended use of the function.

Parameters

- **smoothing** – window size param for smoothing the derivative. It is -1 by default, so it is computed inside the function trying to present an informative smooth TOC. Optional
- **order** – 1 by defautl. 1 uses formulae for derivatives with a cubic error term, and 2, produces a 5-powered error term. Optional

Variables

- **df** – density/TOC derivative function approximation (numpy array). This variable can be publicly acccesed in this TOC.
- **smooth** – smoothed density/TOC derivative (numpy array). This variable can be publicly acccesed in this TOC..
- **areaSm** – area array of the smoothed density (numpy array). The cummulative area of the density, hence it returns for any position the cummulative area from 0 to the postion.
- **areaSm** – area array of the non-smoothed density (numpy array). The cummulative area of the density, hence it returns for any position the cummulative area from 0 to the postion.
- **areaError** – The area and areaSm are forced to integrate 1, nevertheless the numeric approximation actually do not integrate 1 due to the various approximations that the process uses, This variable stores the actual integral, hence it gives a measure of the error.
- **areaSmError** – Similar to areaError but for the smooth version of the TOC.
- **FP** – an array of false positives, hence it is possible to plot or to analyze the density vs the FP.

Returns
A TOC curve of `kind` "density", in contrast with other TOC kinds, it includes the variables mentioned above.

Return type
TOC

1.1.15 Computation of the difference of two TOCs

TOC.**diff**(*T2*)

This method computes the difference between the self TOC (the TOC which the method is called from) and the T2 TOC (self-T2). Possibly the best usage with two normlized TOCs, nevertheless the method does not validate such a case. The TOC with the lowest number of data is resampled, to get to TOC with the same number.

Parameters
T2 – the second TOC that is substracted from the TOC the method is called from (self-T2).

Variables

- **kind** – *'diff'*, the resulting TOC is kind 'diff'.
- **ndata** – The number of data in the resulting curve is the greater of the two input TOCs.

Returns
Returns the a new curve with the difference., new memory is allocated.

Return type
TOC

1.1.16 TOC normalization

TOC.**normalize**()

This method scales the axis to the interval [0,1]. The self TOC (the TOC which the method is called from) is normalized, there is not new memory allocation. :return: Returns the modified TOC curve :rtype: TOC

The kind TOC curve is *'normalized'*. The true positives plus false positives count is 1, ntppfp=1, and true positives, TP=1. Nevertheless the basentppfp and basenpos stores the values of the self TOC.

1.1.17 Plotting the TOC curve

TOC.**plot**(*filename=''*, *title='default'*, *TOCname='TOC'*, *kind='None'*, *height=1800*, *width=1800*, *dpi=300*, *xlabel='default'*, *ylabel='default'*)

A generic plot function for all the kind of TOCs. All the parameters are optional. If filename is not given it plots to a window, otherwise it is a png file.

Parameters

- **filename** – Optional. If given it must be a png filename, otherwise the TOC is plotted to a window.
- **title** – Optional, title of the plot.
- **kind** – Optional, a standard TOC can be plotted normalized or in the original axis values.
- **height** – pixels of the height. 1800 by default.
- **width** – pixels of the width. 1800 by default.
- **dpi** – resolution. 300 by default.
- **xlabel** – string.
- **ylabel** – string.

Returns

it does not return anything.

1.1.18 Compute a probability from a rank value

TOC.**rank2prob**(*rank*, *kind='density'*, *indices=None*)

This function computes probability values associated to a rank value. The thresholds array of the density TOC is used for this purpose. Very possibly the rank is the same than those used to compute a standard TOC instantiated by TOC(rank,groundtruth), hence the indices used in the constructor are available and can save computational time. Otherwise the indices are recomputed. In any case the inputed rank array must be in the same interval than the thresholds.

Parameters

- **rank** – A numpy array with the ran. The intended uses is that this rank comes from the standard TOC computation, and to associate this rank with geolocations, hence the probabilities can be associated in the same order.
- **kind** – if kind is ''density" the probabilitites are computed with the non-smoothed density function, otherwise the smooth version is used. Notice that a this function only must be called from a ''density" kind TOC. Optional

- **indices** – Indices of the reversely sorted rank, this array is computed by the standard TOC computation. Hence the computational cost of recomputing them could be avoided, otherwise the indices are recomputed and they are not stored. Optional

return: a numpy array with the probabilities. The probabilities do not sum 1, instead they sum PDataProp, that is to say they sum the proportion of positives in the data. That is an estimation of the probability of having a 1-class valued datum.

Return type
numpy array

1.1.19 Increasing the TOC data by interpolation

TOC.**resample**(*T2*)

This function creates a new TOC curve, the data of the self TOC (the TOC which the method is called from) is interpolated to the number of data, and the true positves plus false positves, TPplusFP in T2. Considering that the number of true positives plus false positives is modified (usually extended to a greater value)

Parameters
T2 – the TOC curve with the desired number of data and data values in the TPplusFP axis.

Returns
A TOC of the same kind than T2.

Return type
TOC

1.1.20 Vector representation of a TOC

TOC.**vector**(*nmeans=-1*)

Computes a vector representation of the curve using the mean of sets of points. The points are those in a set of intervals with equal number of points. if nmeans=30, then the number of points in the original TOC is divided by 30. If nmeans=-1 (default) the number of intervals is automatically computed, the maximum possible are 8000 the minimum are 1000 or n/5 (the lower).

Parameters

- **nmeans** – The number of intervals. Optional parameter that can be automatically computed. Optional
- **kind** – If the input is a normalized TOC the output is a *'nvector'*, otherwise it is a *'verctor'* kind.
- **basenpos,basentppfp** (*npos, ntppfp,*) – The counts are taken from the input TOC.

Returns
A TOC instance of the vector representation, new memory is allocated.

Return type
TOC

1.2 TOCData class

This class is useful to translate longitude and latitud coordinates and values to a raster object (a rasterio file). The intende use is to translate probabilities from the TOC density to a georeferenced raster object.

class tocdata.**TOCData**(*feature*, *lat*, *lon*, *crs=32641*, *rfile='raster.tif'*)

This class can be used to translate lon, lat and feature values to a raster file, and a raster matrix. The intended use it to translateprobabilities or values from a TOC that are assocaited with lat,lon coordinates, to a raster for visualization.

Parameters

- **feature** – numpy array of feature or probability values.
- **lat** – latitude coordinate associated, index-wise, to a feature value.
- **lon** – longitude coordinate associated, index-wise, to a feature value.
- **rfile** – name of the raster file that is created during the process.

Returns
TOCData object.

Return type
TOCData

1.2.1 Pixel size in latitude and longitude units

TOCData.**DeltaLat = None**

Size of a pixel in the latitude axes.

TOCData.**DeltaLon = None**

Size of a pixel in the longitude axes.

1.2.2 Dimensions of the raster in latitude and longitude units

TOCData.**lenLat = None**

Length of the raster in the latitude axes.

TOCData.**lenLon = None**

Length of the raster in the longitude axes.

1.2.3 Bounding box of the raster in latitude and longitude units

TOCData.**maxLat = None**

Maximum latitude coordinate.

TOCData.**maxLon = None**

Maximum latitude coordinate.

TOCData.**minLat = None**

Minimum latitude coordinate.

TOCData.**minLon = None**

Minmum longitude coordinate.

1.2.4 Number of rows and cols of the raster

TOCData.**nrow = None**

Number of rows in the raster.

TOCData.**ncol = None**

Number of cols in the raster.

1.2.5 Raster matrix

TOCData.**raster = None**

Raster matrix .

1.2.6 Raster file

TOCData.**raster_file = None**

Reference to the raster file as a rasterior object.

INDEX