myutils

Release 1.0

Guillermo Díez-Señorans

Dec 14, 2023

CONTENTS:

CHAPTER

ONE

MYUTILS PACKAGE

class myutils.**BarraProgreso**(*total*, *long_barra=40*, *caracter='#'*)

Bases: object

Esta clase instancia un objeto BarraProgreso.

Parameters

- **total** (*int*) – Número de iteraciones que hacen el 100% de la barra.
- **long_barra** (*int, opcional*) – Número de caracteres que componen la barra, por defecto 40.
- **caracter** (*char, opcional*) – Caracter que se utiliza para decorar la barra, por defecto '#'.

Return type
None

myutils.**export_legend_plt**(*ax*, *name='legend.pdf'*, *pad_inches=0.1*, ***kwargs*)

Exporta la leyenda de la figura actualmente activa.

Esta función es útil para exportar la leyenda de un gráfico cuando se tienen múltiples gráficos que comparten la misma leyenda. Es importante tener en cuenta que se requiere haber creado una figura con 'plt.plot', la cual debe incluir una leyenda, y NO se debe haber ejecutado 'plt.show()'.

Parameters

- **ax** (*matplotlib.axes.Axes*) – Los ejes de la figura de los cuales se quiere extraer la leyenda. Si se utilizan comandos como 'plot', etc., se puede obtener los ejes activos del gráfico con ax=plt.gca().
- **name** (*str, opcional*) – Nombre de la imagen. Por defecto es "legend.pdf".
- **pad_inches** (*float, opcional*) – Esta opción se pasa directamente a la función "savefig". Por defecto es 0.1.
- **kwargs** (*dict*) – Las opciones introducidas aquí se pasan directamente a la función *ax.legend*.

Return type
None

myutils.**run_in_parallel**(*func*, *args*)

Ejecuta la función en paralelo utilizando argumentos proporcionados.

Esta función toma un método 'func' y una lista de argumentos 'args', y repite la ejecución de la función en paralelo. El método 'func' puede recibir un número arbitrario y tipos de argumentos, pero serán pasados en orden según aparezcan en 'args'.

Parameters

- **func** (*function*) – Procedimiento que puede tomar un número arbitrario y tipos de argumentos, pero son específicos en su posición.
- **args** (*list*) – Lista que contiene una lista con los argumentos a pasar a 'func' en cada proceso.

Returns
Lista con los resultados de cada proceso.

Return type
list

Warning: Esta función siempre debe ser ejecutada dentro de *if __name__=='__main__':*.

myutils.**set_size_plt**(*ax*, *x*, *y*)

Esta función permite dibujar una figura fijando el tamaño del plot, y no de la figura completa. Toma los valores *x*, *y* que representan las dimensiones en dichos ejes de un plot, y construye la figura del tamaño que sea necesario para acomodar los ejes.

Parameters

- **ax** (*matplotlib.axes.Axes*) – Objeto que contiene la figura a redimensionar.
- **x** (*float*) – Nueva anchura.
- **y** (*float*) – Nueva altura.

Return type
None

1.1 Submodules

1.2 myutils.bool module

class myutils.bool.**LinearSystECC**(*dim=4*, *length=7*, *G=False*, *H=False*)

Bases: object

Clase para manipular sistemas de códigos de corrección de errores lineales.

Esta clase proporciona funcionalidades para la generación de matrices G y H, codificación y decodificación de vectores para códigos de corrección de errores lineales en un cuerpo finito.

Parameters

- **dim** (*int, opcional*) – Dimensión de la matriz. Por defecto 4.
- **length** (*int, opcional*) – Longitud de la matriz. Por defecto 7.
- **G** (*list of list of bool, opcional*) – Matriz booleana G para operaciones de generación de código. Por defecto False.
- **H** (*list of list of bool, opcional*) – Matriz booleana H para operaciones de generación de código. Por defecto False.

Raises
ValueError – Si la matriz generadora y la matriz de paridad no cumplen GH^=0.

Notes

Esta clase permite realizar operaciones en un cuerpo finito para códigos de corrección de errores lineales. Los métodos 'encode' y 'decode' permiten la codificación y decodificación de vectores respectivamente, usando las matrices generadora y de paridad.

decode(*vector*)

Decodifica un código de entrada de 'length' bits en un mensaje de 'dim' bits utilizando un método de mínima distancia de Hamming.

Este método es absolutamente ineficiente y no debería usarse con dimensiones mayores que 9.

Parameters
vector (*list of bool*) – Código booleano de entrada a decodificar.

Returns
Mensaje decodificado.

Return type
list of bool

Raises
ValueError – Si la longitud del vector de entrada no coincide con 'length'. Si hay demasiados errores para decodificar el vector de manera fiable.

encode(*vector*)

Codifica un mensaje de entrada de 'dim' bits en un código de 'length' bits.

Parameters
vector (*list of bool*) – Mensaje booleano de entrada a codificar.

Returns
Mensaje booleano codificado.

Return type
numpy.ndarray

Raises
ValueError – Si la longitud del vector de entrada no coincide con 'dim'.

myutils.bool.**gauss_elimination_gf2**(*coeff_matrix_in*, *const_vector_in*)

Resuelve un sistema de ecuaciones booleanas en un cuerpo finito con las operaciones XOR ("+") y AND ("*").

Parameters

- **coeff_matrix_in** (*list of list of bool*) – Matriz booleana que representa los coeficientes del sistema de ecuaciones.
- **const_vector_in** (*list of bool*) – Vector booleano que representa el término independiente de cada ecuación.

Returns
Matriz booleana triangular.

Return type
numpy.ndarray

myutils.bool.**hamming**(*in1*, *in2*)

Calcula la distancia de Hamming (medida en bits) entre dos entradas.

Parameters

- **in1** (*list of bool*) – Primer vector booleano de entrada.

- **in2** (*list of bool*) – Segundo vector booleano de entrada.

Returns
Distancia de Hamming entre las dos entradas (medida en bits).

Return type
int

myutils.bool.**invert_matrix_gf2**(*matrix_in*)

Devuelve la inversa de una matriz booleana en un cuerpo finito con las operaciones XOR ("+") y AND ("*").

Parameters
matrix_in (*list of list of bool*) – Matriz booleana a invertir.

Returns
Matriz booleana inversa.

Return type
numpy.ndarray

myutils.bool.**matrix_multiply_gf2**(*matrix1_in*, *matrix2_in*)

Multiplica un par de matrices booleanas en un cuerpo finito con las operaciones XOR ("+") y AND ("*").

Esta función puede utilizarse también para multiplicar vectores por matrices, gestionando automáticamente la disposición del vector: fila si va delante de la matriz, columna si va detrás. Así, si 'matrix1_in' y 'matrix2_in' son dos vectores, el resultado será el producto vectorial.

Parameters

- **matrix1_in** (*list of list of bool*) – Primera matriz booleana.
- **matrix2_in** (*list of list of bool*) – Segunda matriz booleana.

Returns
Producto de las matrices o el producto vectorial en gf(2).

Return type
numpy.ndarray

1.3 myutils.stats module

myutils.stats.**Dks_montecarlo_discrete**(*model*, *fit*, *N*, *verbose=True*, ***kwargs*)

Calcula la distribución del estadístico KS de un modelo discreto en comparación con una distribución teórica.

Parameters

- **model** (*callable*) – Función aleatoria a contrastar. El primer parámetro que debe aceptar es 'N', seguido de un número arbitrario de parámetros que son pasados mediante 'kwargs'. Devuelve un array de valores aleatorios.
- **fit** (*list of float*) – Lista de valores que representa la distribución de probabilidad contra la cual se calcula la distribución de KS.
- **N** (*int*) – Número de veces que se repite el cálculo de Dks.
- **verbose** (*bool, optional*) – Si es True, imprime el progreso. Por defecto es True.
- **kwargs** (*dict*) – Parámetros a pasar a 'model'.

Returns
Lista de valores Dks.

Return type
list of float

Notes

Esta función calcula la distribución del estadístico KS de un modelo 'model' que produce 'N' valores aleatorios frente a una curva teórica 'fit', representada como una lista de valores. Devuelve una lista de estos índices KS.

El método 'model' admite una cantidad arbitraria de parámetros pasados mediante 'kwargs'.

myutils.stats.**bin_rv_cont**(*rv_continuous*, *bins_in*, **args*)

Calcula la probabilidad acumulada de un objeto 'rv_continuous' de scipy en los extremos dados por 'bins'.

Parameters

- **rv_continuous** (*scipy.stats.rv_continuous*) – Objeto 'rv_continuous' de scipy.
- **bins_in** (*array_like*) – Bineado del eje de abscisas. Debe ser una lista de 'N_bins+1' elementos, conteniendo los extremos de cada bin.
- **args** (*list, optional*) – Parámetros extra para pasar a 'rv_continuous.cdf', distintos del primer argumento ('x').

Returns
Arreglo de tamaño 'N_bin' elementos que contiene la probabilidad acumulada en los extremos dados por 'bins'. El arreglo de salida está normalizado respecto a la suma de sus elementos.

Return type
numpy.ndarray

Notes

Esta función toma un objeto 'rv_continuous' de scipy y una lista 'bins_in' que contiene el bineado del eje de abscisas. Calcula la probabilidad acumulada en los extremos dados por 'bins' utilizando 'rv_continuous.cdf' y devuelve un numpy array normalizado con las probabilidades acumuladas.

El argumento 'args' se utiliza para pasar parámetros extra distintos al primero ('x') a 'rv_continuous.cdf'.

myutils.stats.**bin_rv_discrete**(*rv_discrete*, *bins_in*, **args*)

Calcula la probabilidad acumulada de un objeto 'rv_discrete' de scipy en los extremos dados por 'bins'.

Parameters

- **rv_discrete** (*scipy.stats.rv_discrete*) – Objeto 'rv_discrete' de scipy.
- **bins_in** (*array_like*) – Bineado del eje de abscisas. Debe ser una lista de 'N_bins+1' elementos, conteniendo los extremos de cada bin.
- **args** (*list, optional*) – Parámetros extra para pasar a 'rv_discrete', distintos del primer argumento ('k').

Returns
Arreglo de tamaño 'N_bin' elementos que contiene la probabilidad acumulada en los extremos dados por 'bins'. El arreglo de salida está normalizado respecto a la suma de sus elementos.

Return type
numpy.ndarray

Notes

Esta función toma un objeto 'rv_discrete' de scipy y una lista 'bins_in' que contiene el bineado del eje de abscisas. Calcula la probabilidad acumulada en los extremos dados por 'bins' y devuelve un numpy array normalizado con las probabilidades acumuladas.

El argumento 'args' se utiliza para pasar parámetros extra distintos del primero ('k') a 'rv_discrete'.

myutils.stats.**chisq_gof_discrete**(*obs_data*, *model*, *alpha=0.05*, *plothist=False*, ***kwargs*)

Realiza un test chi^2 de bondad del ajuste entre los datos observados y un modelo estocástico discreto (una función pmf -probability mass function-) que hipotéticamente genera dichos datos.

Parameters

- **obs_data** (*array_like*) – Vector de datos observados.
- **model** (*callable*) – Función pmf (probability mass function) que representa el modelo estocástico discreto. El primer argumento siempre debe ser la variable independiente (aleatoria).
- **alpha** (*float, optional*) – Nivel de significancia para el test chi^2. Por defecto es 0.05.
- **plothist** (*bool, optional*) – Indica si se debe graficar el histograma de los datos. Por defecto es False.
- **kwargs** (*dict*) – Opciones adicionales para pasar parámetros a la función pmf introducida como modelo.

Returns

Un diccionario con los campos: - 'chisq': Valor del estadístico chi^2. - 'pvalue': Valor tal que la probabilidad de obtener 'chisq' es al menos pvalue. - 'dof': Número de grados de libertad de la distribución chi^2. - 'testpass': Booleano que indica si el test ha sido superado en función de 'alpha'.

Return type

dict

Notes

Esta función calcula un test chi^2 de bondad del ajuste entre los datos observados y un modelo estocástico discreto proporcionado como una función pmf. Se genera un histograma de los datos y se calculan los valores esperados de probabilidad para cada bin. Luego se aplica la función 'chisquare' de scipy para realizar el test chi^2.

El parámetro 'kwargs' se puede utilizar para pasar parámetros adicionales a la función pmf del modelo.

myutils.stats.**fit_rv_cont**(*data*, *rv_cont*, *bins=10*, *alpha=0.05*, *plot=False*)

Encuentra la distribución 'rv_continuous' que mejor ajusta un conjunto de datos 'data' y calcula el 'valor p' correspondiente a una significancia 'alpha'.

Parameters

- **data** (*list of float*) – Vector con los datos a ajustar.
- **rv_cont** (*scipy.stats.rv_continuous*) – Objeto 'scipy.stats.rv_continuous' que define una distribución de probabilidad continua que se ajustará a los datos 'data'.
- **bins** (*int, optional*) – Número de cajas para el histograma de 'data'. Por defecto es 10.
- **alpha** (*float, optional*) – Valor de significancia 'alpha' para el cual se calcula el valor p. Por defecto es 0.05.

- **plot** (*bool, optional*) – Si es True, pinta el histograma y superpone la curva encontrada. La función no ejecuta 'show()', de forma que el usuario puede recabar la figura externamente con 'gca()' y 'gcf()', y editarla antes de representarla. Por defecto es False.

Returns

Una lista que contiene tres elementos: - El primero es una lista de los parámetros que mejor ajustan la distribución (en el mismo orden en que se definen en la función 'rv_cont.pdf'). - El segundo es un nparray con el histograma. - El tercero es un nparray con el bineado de los datos de entrada.

Return type

list

Notes

Esta función es un wrapper para encontrar la distribución 'rv_continuous' que mejor ajusta un conjunto de valores 'data' utilizando la máxima verosimilitud. Calcula el 'valor p' correspondiente a una significancia 'alpha'. Resulta útil para distribuciones obtenidas por simulación, que carecen de una función densidad teórica.

myutils.stats.**get_area_fraction**(*hist*, *p=0.9*)

Encuentra los extremos 'xmin' y 'xmax' de una lista 'hist' de números (no necesariamente normalizada) de tal manera que el intervalo 'hist[xmin:xmax]' contenga al menos una fracción 'p' de la suma acumulada de los elementos de la lista.

Parameters

- **hist** (*list or array_like*) – Lista de números.
- **p** (*float, optional*) – Fracción mínima deseada. Por defecto es 0.9.

Returns

- **xmin** (*int*) – Índice del extremo izquierdo del intervalo.
- **xmax** (*int*) – Índice del extremo derecho del intervalo.

Notes

Esta función busca los extremos 'xmin' y 'xmax' en la lista 'hist' de números, de tal manera que el intervalo 'hist[xmin:xmax]' contenga al menos una fracción 'p' de la suma acumulada de los elementos de la lista. La búsqueda comienza desde el índice correspondiente al valor máximo de la lista y se desplaza simétricamente hacia ambos extremos.

myutils.stats.**get_hist_smooth**(*data*, *p=0.9*, *x0=nan*, *x1=nan*)

Genera un histograma suavizado a partir de un conjunto de datos 'data'.

Parameters

- **data** (*array_like*) – Conjunto de datos para representar en forma de histograma.
- **p** (*float, optional*) – Fracción (tanto por uno) de área bajo la cual se busca que el histograma no presente máximos relativos. Por defecto es 0.9.
- **x0** (*float, optional*) – Extremo inferior del rango del histograma. Por defecto es NaN (se calcula automáticamente a partir de 'data').
- **x1** (*float, optional*) – Extremo superior del rango del histograma. Por defecto es NaN (se calcula automáticamente a partir de 'data').

Returns
Histograma suavizado generado.

Return type
numpy.ndarray

Notes

Esta función devuelve un histograma con el número de bins adecuado para que no presente máximos locales, lo cual resulta en un aspecto más suave. El parámetro 'p' representa la fracción de área bajo la cual se busca que el histograma no presente máximos relativos, medido simétricamente desde el punto máximo del histograma.

Los parámetros 'x0' y 'x1' son opcionales y se utilizan para definir el rango del histograma; si no se proporcionan, se calculan automáticamente a partir de los valores mínimo y máximo de 'data'.

myutils.stats.**intervalo_int**(*inf*, *sup*)

Esta función auxiliar devuelve los extremos de números enteros dado un intervalo real.

Parameters

- **inf** (*float*) – Extremo inferior del intervalo.
- **sup** (*float*) – Extremo superior del intervalo.

Returns
Rango de enteros entre los extremos *inf* y *sup*.

Return type
list

1.4 myutils.tensor module

Este módulo permite definir nombres para los ejes de un numpyarray.

class myutils.tensor.**Tensor**(*array*, *axis=False*)

Bases: object

Un objeto que es esencialmente un nparray junto con una variable 'self.axis' que permite indexar los ejes del array mediante un nombre natural, en lugar de un índice entero.

Parameters

- **array** (*numpy.ndarray*) – Arreglo que contiene los datos y la forma del tensor.
- **axis** (*list of strings, optional*) – Nombres de los ejes. El orden en que se proporcionan los nombres se correlaciona con el correspondiente eje del parámetro 'array'. Si no se introduce una lista de nombres en esta variable, se asignarán por defecto como nombres el índice de cada eje (empezando en '0').

array

Arreglo que contiene los datos y la forma del tensor.

Type
numpy.ndarray

shape

Forma del tensor (igual que en un nparray).

Type
tuple

`axis`

Nombres de los ejes. El i-ésimo elemento contiene el nombre del i-ésimo eje. Si no se proporciona, se asignan nombres predeterminados en forma de índices.

Type
list of strings

`item`(***axes*)

Esta función toma por argumento el nombre de cada eje igualado a un índice, y devuelve el valor guardado en esa coordenada.

Parameters
axes (*dict*) – Argumentos clave-valor donde las claves son los nombres de los ejes y los valores son los índices para esos ejes.

Returns
Valor almacenado en las coordenadas especificadas.

Return type
object

`mean`(*axe*)

Devuelve un Tensor comprimido en el eje 'axe', suprimiéndolo y calculando el promedio a lo largo de dicho eje.

Parameters
axe (*int*) – Índice del eje a comprimir y calcular el promedio.

Returns
Tensor con el eje 'axe' suprimido y cada punto contiene el promedio a lo largo de ese eje.

Return type
Tensor

`save`(*file_name*, *fmt='%.18e'*)

Guarda el Tensor en un archivo de texto.

Parameters

- **file_name** (*str*) – Nombre del archivo donde se guardará el Tensor.
- **fmt** (*str, optional*) – Formato numérico de los datos. Sigue el mismo formato que el parámetro 'fmt' de numpy.savetxt.

Return type
None

`size`(*axis*)

Esta función toma el nombre de un eje y devuelve su dimensión.

Parameters
axis (*str*) – Nombre del eje del que se desea obtener la dimensión.

Returns
Dimensión del eje especificado.

Return type
int

slice(**axes*)

Toma una serie de ejes (por nombre), igualados a una porción 'slice'. Esta porción puede ser un entero, una lista de dos enteros, o una lista de tres enteros, y devuelve un Tensor reducido al 'slice' indicado (usando la misma notación que un slice estándar en Python: [x] es la x-ésima entrada, [x,y] son todas entradas en [x,y), y [x,y,z] son todas las entradas en [x,y) tomadas en saltos de z). Esta función es en realidad un wrapper a 'take', usando la nomenclatura de Python para hacer slicing.

Parameters
axes (*dict*) – Argumentos clave-valor donde las claves son los nombres de los ejes y los valores son objetos 'slice' o listas de enteros para realizar la porción en esos ejes.

Returns
Tensor reducido al 'slice' indicado a lo largo de los ejes especificados.

Return type
Tensor

squeeze()

Elimina todos los ejes del tensor cuya dimensión sea igual a '1'.

Returns
Tensor resultante después de eliminar los ejes con dimensión '1'.

Return type
Tensor

std(*axe*)

Devuelve un Tensor comprimido en el eje 'axe', suprimiéndolo y calculando la desviación estándar a lo largo de dicho eje.

Parameters
axe (*int*) – Índice del eje a comprimir y calcular la desviación estándar.

Returns
Tensor con el eje 'axe' suprimido y cada punto contiene la desviación estándar a lo largo de ese eje.

Return type
Tensor

swap(*axe1*, *axe2*)

Devuelve un Tensor con el mismo "array" pero los ejes "axe1" y "axe2" intercambiados.

Parameters

- **axe1** (*int*) – Índice del primer eje a intercambiar.
- **axe2** (*int*) – Índice del segundo eje a intercambiar.

Returns
Tensor con los ejes "axe1" y "axe2" intercambiados.

Return type
Tensor

take(**axes*)

Toma una serie de ejes (por nombre), igualados a una lista de índices, y devuelve un Tensor reducido a los índices seleccionados.

Parameters
axes (*dict*) – Argumentos clave-valor donde las claves son los nombres de los ejes y los valores son listas de índices a lo largo de esos ejes.

Returns
Tensor reducido a los índices seleccionados a lo largo de los ejes especificados.

Return type
Tensor

myutils.tensor.**append**(*axe*, **args*)

Concatena dos objetos 'Tensor' a lo largo de un eje especificado por su nombre.

Parameters

- **axe** (*str*) – Nombre del eje a lo largo del cual se realizará la concatenación.
- **args** (Tensor) – Tensores que se concatenarán. Se deben proporcionar al menos dos tensores de entrada. Todos los tensores deben tener un parámetro 'Tensor.axis' idéntico y las mismas dimensiones en todos los ejes excepto posiblemente en 'axe'.

Returns
Nuevo Tensor que consiste en la concatenación de los argumentos en el orden proporcionado.

Return type
Tensor

Raises
ValueError – Si no se cumplen los requisitos de entrada (al menos dos tensores y 'axe').

myutils.tensor.**load_tensor**(*file_name*)

Devuelve un Tensor almacenado en un archivo de texto.

Parameters
file_name (*str*) – Nombre del archivo que contiene el Tensor.

Returns
Tensor cargado desde el archivo de texto.

Return type
Tensor

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m