Cheatography

Python 1.1 Numpy Cheat Sheet

by julenx via cheatography.com/168626/cs/35314/

Creación y copia de arrays

np.array([[list1, list2, list3], [list4, list5,lis6]])

Creación de array 2x3. Podemos forzar el tipo del array devuelto con el parámetro dtype (dtype = "int")

a.ndim

número de dimensiones del array

a.shape

Dimensiones y elementos en cada dimensión

a.dtype

tipo del array

np.arange(start, stop, step)

Genera array de un conjunto de números entre un valor de inicio y uno final. Si no especificamos el tercer argumento, se sobreentiende que el incremento es de 1. Si solo añadimos un valor como argumento, la función considera todos los valores desde el cero hasta dicho valor (sin incluirlo).

np.linspace(start, finish, n)

genera un array NumPy formado por n números equiespaciados entre dos dados. el array generado incluye el valor final. podemos especificar el tipo del array usando el parámetro dtype:

np.logspace(start, finish, n)

en una escala logarítmica, o, dicho con otras palabras, devuelve también el número de valores especificado (como en linspace), pero devolviendo el resultando de elevar 10 a dichos valores

Creación y copia de arrays (cont)

np.empty(shape = (r, c))

crea un array de las dimensiones indicadas pero sin inicializar sus valores. Si solo indicamos una dimensión... el array será unidimensional. Si se muestra, saldrán números aleatorios.

np.zeros(shape = (r, c))

np.ones(shape = (r, c))

np.full(shape = (r, c), fill_value = n)

crea un array de las dimensiones indicadas, rellenándolo con el valor que se especifique

np.empty_like(a)

genera un array sin inicializar con las mismas dimensiones y tipo que otro array dado.

np.zeros_like(a)

np.ones_like(a)

np.full_like(a, fill_value = n)

np.eye(r, c)

genera un array de dos dimensiones (con el tamaño indicado por los dos primeros argumentos) con ceros salvo en la diagonal principal, donde se insertan unos.

np.identity(d)

genera una matriz identidad: matriz cuadrada de ceros salvo en la diagonal principal.

np.repeat(a, times, axis = n)

replica los elementos de un array a lo largo del eje indicado:

a.astype("int")

hacer un copia de un array aplicándole otro tipo de datos

Creación y copia de arrays (cont)

b = a.copy()

realizar una copia independiente de un array

b = a.astype("int")

hacer un copia de un array aplicándole otro tipo de datos

Permutaciones

np.random.permutation(list/array)

devuelve un array con la versión desordenada de la secuencia cedida como argumento. Si el array indicado tiene más de una dimensión, solo se desordena a lo largo del primer eje.

np.random.shuffle(list/array)

modifica una secuencia in-place desordenando sus elementos

Funciones de conjuntos

np.unique(a)

devuelve un array ordenado con los elementos únicos de las estructuras de entrada

np.union1d(a, b)

devuelve un array unidimensional ordenado con los elementos resultantes de unir los arrays indicados

np.intersect1d(a, b)

devuelve un array ordenado con los valores únicos de la intersección de las estructuras de entrada (devuelve los elementos que se repitan en ambos)

Funciones de conjuntos (cont)

np.intersect1d(a, b)

devuelve un array ordenado con los valores únicos de la intersección de las estructuras de entrada (devuelve los elementos que se repitan en ambos)

np.setdiff1d(a, b)

Return the unique values in a that are not in b

np.setdiff1d(a, b)

Return the unique values in a that are not in b

np.setxor1d(a, b)

devuelve un array ordenado con los valores únicos del conjunto de elementos que pertenecen a a o a b, pero no a ambos

np.in1d(a, b)

devuelve un array ordenado unidimensional booleano con los elementos de a que se encuentran también en b

np.isin(a, b)

devuelve un array de las mismas dimensiones que el array a en el que se indique con booleanos si el elemento correspondiente está o no incluido en b

Lectura y escritura de ficheros

np.save("my_array", a)

guardar el contenido de un array en disco. Con esta función los ficheros se graban con extensión ".npy". Si no indicamos esta extensión, se añadirá automáticamente (si indicamos otra extensión, la extensión .npy se añadirá tras la nuestra)

Lectura y escritura de ficheros (cont)

a = np.load("array.npy")

Cargar fichero

np.savez("arrays", a, b)

guardar varios arrays NumPy en el mismo archivo. El fichero se guarda con la extensión ".npz" (la función de lectura sigue siendo la misma: numpy.load). Al leer el fichero guardado, se devuelve un objeto con estructura de lista cuyos elementos contienen los arrays individuales.

arrays.files

comprobar la lista de índices de una lista de arrays. Luego se podrá acceder con arrays["nombre"]

np.savetxt("array.txt", a)

Guardar array como archivo txt

a = np.loadtxt("array.txt")

Cargar array desde archivo txt

np.savetxt("array.csv", a, delimiter = ",")

Guardar array como archivo csv

a = np.loadtxt("array.csv", delimiter = ",")

Cargar array desde archivo csv

Funciones de álgebra lineal

np.dot(a, b)

devuelve el producto escalar de dos arrays

a.dot(b)

np.linalg.det(a)

devuelve el determinante de la matriz representada por el array

np.linalg.inv(a)

devuelve la inversa de la matriz. Se puede comprobar con np.round(a.dot(-np.linalg.inv(a)))

La función where

np.where(c, x, y)

acepta como argumentos de entrada una condición y dos estructuras tipo array, x e y, y devuelve valores de x o de y en función de que se cumpla o no la condición: Si se cumple, se devuelve el valor de x. Si no se cumple, el de y. Por ejemplo, en la posición de c escribimos x > 2

Redimensionamiento

a[(a >= 6) | (a == 2)] = 0

Edición con booleanos

a.shape = (4, 3)

fijar el tamaño del array. También con esta función es posible utilizar el valor -1 para dejar que sea Numpy quien calcule el tamaño adecuado

b = np.reshape(a, newshape = (4, 3))

devuelve un nuevo array con los datos del array cedido como primer argumento y el nuevo tamaño indicado:

b = a.reshape((4, -1))

a.flatten()

devuelve una copia del array colapsado a una única dimensión

b = np.transpose(a)

Transposición de arrays

c = a.transpose()

a.T

Operaciones con arrays

a // 2

Redondeo al entero más próximo

np.multiply(a, b)

a*b

np.add(a, b)

a+b

np.subtract(a, b)

a-b

np.divide(a, b)

a/2

np.power(a, b)

a**b

np.rint(a)

Redondea sus elementos al entero más próximo. no preserva el dtype del array (float64)

np.sign(a)

Devuelve un array NumPy del mismo tamaño con valores que indican el signo de los elementos del array de entrada (1 si el elemento es positivo, 0 si es cero, y -1 si el elemento es negativo)

np.exp(a)

Calcula la exponencial de los elementos de la estructura de entrada. e^x

np.log(a)

Logaritmo neperiano o natural de los elementos de la estructura de entrada:

Operaciones con arrays (cont)

np.sqrt(a)

Raíz cuadrada de los valores de la estructura de entrada. si se encuentra un valor negativo y se está trabajando en el dominio de los números reales, se devuelve un error. Esto no ocurre en el dominio de los números imaginarios (dtype="complex")

np.square(a)

Devuelve el cuadrado de los elementos de la estructura de entrada

np.gcd(a, b)

Devuelve el máximo común divisor de los elementos de las dos estructuras de entrada (dos arrays o un array y un número)

np.lcm(a, b)

Mínimo común múltiplo de los elementos de las dos estructuras de entrada (dos arrays o un array y un número)

np.sin(a)

Devuelve el seno de los elementos

np.cos(a)

Coseno de los elementos

np.tan(a)

Tangente de los elementos

np.arcsin(a)

Arcoseno de los elementos

np.arccos(a)

Arcocoseno de los elementos

np.arctan(a)

Arcotangente de los elementos

Operaciones con arrays (cont)

np.deg2rad(a)

Convierte ángulos de grados sexagesimales a radianes

np.rad2deg(a)

Convierte ángulos de radianes a grados sexagesimales:

np.bitwise_and(a, b)

Aplica un "y-lógico" a nivel de bit

Funciones para arrays booleanos

(a > 0).mean()

porcentaje de elementos mayores que cero

a.all()

devuelve True cuando todos los valores a lo largo del eje indicado del array son True. Si no se especifica el eje, se aplana el array. En un array numérico los números diferentes a cero se interpretan como True, y aquellos con valor 0, como False. cuando el array está vacío, el método devuelve el valor lógico True

a.any()

devuelve True cuando algún valor a lo largo del eje indicado del array es True. Si el array está vacío, devuelve 0

Aceptan el parámetro axis

Funciones polinómicas

p = np.polynomial.polynomial.Polynomial([1, -5, 1, -2])

Representar el polinomio $-2x^3 + x^2 - 5x + 1$

list(p)

extraer los coeficientes como lista

Funciones polinómicas (cont)

p.coef

extraer los coeficientes como array

p.degree()

extraer el grado del polinomio

p(x)

Evaluación de un polinomio (clase Polynomial)

np.polynomial.polynomial.polyval(list, c)

evaluación para el punto (o los puntos) del polinomio con lista de coeficientes.

p.deriv(n)

Derivación de polinomios (clase polinomial). La n indica el orden de la derivada (primera, segunda, tercera...) Si no se especifica, se calcula la primera derivada.

np.polynomial.polynomial.polyder(c, n)

Derivación de polinomios con lista de coeficientes. El segundo argumento indica el orden de la derivada (primera, segunda, tercera...) Si no se especifica, se calcula la primera derivada.

p.integ()

Integración de polinomios. Se puede cambiar el valor de la constante k incluyendo el parámetro k=valor. Se puede calcular varias veces con el parámetro m=valor. En este caso, si se incluye el valor k este puede ser una lista cuyos elementos se asignaran en orden a cada integral.

np.polynomial.polynomial.polyint(c)

Calcular integral con lista de coeficientes. También acepta k y m

p.roots()

Raíz de un polinomio

Funciones polinómicas (cont)

np.polynomial.polynomial.polyroots(c)

Raíz de un polinomio si estamos trabajando solo con los coeficientes del polinomio

f = np.polynomial.polynomial.Polynomial.fit(x, y, n, domain = (-1, 1))

devuelve un objeto de tipo Polynomial representando el polinomio del grado indicado que minimiza la suma de cuadrados de las diferencias en las ordenadas (método de mínimos cuadrados). La n es el grado del polinomio

f = np.polynomial.polynomial.polyfit(x, y, n)

nos permite aproximar un conjunto de datos según el procedimiento de mínimos cuadrados devolviendo un array NumPy con los coeficientes del polinomio. La n es el grado del polinomio.

p = np.polynomial.polynomial.Polynomial.fromroots(coef_list)

acepta como argumento una lista de raíces (puede ser una lista Python o una estructura semejante) y devuelve el polinomio correspondiente

c = np.polynomial.polynomial.polyfromroots(coef_list)

acepta la misma lista de raíces y devuelve los coeficientes del polinomio resultante

polyadd(c1, c2)

suma dos polinomios

polysub(c1, c2)

resta dos polinomios

polymul(c1, c2)

resta dos polinomios

Funciones polinómicas (cont)

polymulx(c)

multiplica un polinomio por la variable independiente x

polydiv(c1, c2)

divide un polinomio por el otro

polypow(c, pow)

eleva un polinomio a una potencia

Hay dos formas de trabajar con polinomios: La primera consiste en representar el polinomio utilizando la clase Polynomial que podemos encontrar en la sublibrería numpy.polynomial.polynomial). Esta clase proporciona los métodos estándar +, -, *, //, %, divmod,* y (), así como otros métodos y atributos.

La segunda forma de trabajar con polinomios es representarlos por medio de un array (o estructura semejante) de coeficientes.

Ordenación de arrays

np.sort(a)

Podemos ordenar un array mediante la función numpy.sort, función que permite especificar el eje (o dimensión) por el que se desea realizar la ordenación. Si el parámetro correspondiente se fija al valor None, el array es aplanado antes de ordenarlo. Por defecto, este parámetro toma el valor -1, indicando que la ordenación se va a realizar a lo largo del último eje.

a.sort(axis)

La ordenación se produce in-place. la opción de axis = None no está permitida

Distribuciones

a = np.random.normal(loc = m, scale = sd, size = n)

genera un array del tamaño indicado a partir de una distribución normal o gaussiana de una cierta media y desviación estandar.

numpy.random.beta(a, b, size=None)

random.chisquare(df, size=None)

random.exponential(scale=1.0, size=None)

random.poisson(lam=1.0, size=None)

Números aleatorios

np.random.seed(seed = n)

inicializar el generador de números aleatorios para que de los mismos resultados cada vez que se ejecute el código

a = np.random.rand(r, c)

genera un array del tamaño indicado conteniendo números aleatorios extraídos del intervalo [0, 1) a partir de una distribución uniforme. Si se solo se indica el primer parámetro, se genera un array plano

a = np.random.random(size = (r, c))

genera un array del tamaño indicado conteniendo números reales aleatorios extraídos de una distribución continua uniforme en el intervalo [0.0, 1.0):

Números aleatorios (cont)

a = np.random.randint(s, f, size = (r, c))

genera un array del tamaño indicado conteniendo números enteros aleatorios extraídos de una distribución discreta uniforme entre los intervalos dados (desde un valor inferior incluido, hasta un valor superior sin incluir)

a = np.random.choice(array, n)

devuelve un array del tamaño indicado conteniendo una muestra del array unidimensional cedido como argumento. El parámetro replace determina si la selección se hace o no con reemplazo (por defecto este parámetro toma el valor True)

Funciones matemáticas

np.mean(a)

Devuelve la media aritmética de los valores del array a lo largo de los ejes especificados. Si no se especifica, se calcula la media con todos los valores. También se pueden especificar capas (axis=(0, 2))

np.median(a)

Devuelve la mediana de los valores de un array a lo largo de los ejes especificados. Mismas opciones que mean

np.std(a)

Devuelve la desviación estándar de los valores un array a lo largo de los ejes especificados

np.var(a)

Devuelve la varianza de los valores de un array a lo largo de los ejes especificados

Funciones matemáticas (cont)

np.sum(a)

devuelve la suma de los valores de un array a lo largo de los ejes especificados

np.cumsum(a)

devuelve la suma acumulada de los valores de un array a lo largo de los ejes especificados

np.corrcoef(a)

devuelve el coeficiente de correlación de Pearson entre las filas de un array bidimensional o entre dos arrays unidimensionales np.corrcoef(a, b)

np.average(a, weights = w)

Devuelve la media aritmética ponderada de los valores de un array a lo largo de los ejes especificados. Los pesos a utilizar se indican con el parámetro weights en forma de lista. Si no se especifican los pesos, la función asume que todos los valores sobre los que se va a aplicar el cálculo tienen idéntico peso:

np.argmax(a)

devuelve los índices de los mayores valores del array a lo largo del eje indicado. Si no se especifica el eje, se aplana el array.

np.argmin(a)

devuelve los índices de los menores valores del array a lo largo del eje indicado. Si no se especifica el eje, se aplana el array.

Funcionan lo largo de los ejes especificados (axis=0). Si no se especifica, se calcula la función con todos los valores. También se pueden especificar capas (axis=(0, 2))

Funciones universales flotantes

np.isnan(a)

evalúa si los elementos de la estructura de entrada son NaN

np.floor(a)

devuelve el mayor entero menor o igual que los elementos

np.ceil(a)

devuelve el menor entero mayor o igual que los elementos

np.trunc(a)

devuelve los elementos de la estructura de entrada truncados. El valor truncado de un número es el entero más próximo al mismo que está más cerca del cero que el propio elemento

np.round(a, 2)

devuelve un array equivalente al cedido como argumento tras sustituir cada elemento por la versión redondeada a un cierto número de decimales. El número de decimales se indica como segundo argumento.

numpy.around

a_function = np.vectorize(function)

Permite crear una función vectorizada a partir de una que no lo es, lo que nos permite aplicar esta última a un array.

Funciones universales de comparación

np.greater(a, b)

valor verdadero de la comparación x1 > x2, comparando elemento a elemento

np.greater_equal(a, b)

valor verdadero de la comparación x1 ≥ x2

np.less(a, b)

valor verdadero de la comparación x1 < x2

np.less_equal(a, b)

valor verdadero de la comparación x1 ≤ x2

np.not_equal(a, b)

valor verdadero de la comparación x1 ≠ x2

np.equal(a, b)

valor verdadero de la comparación x1 == x2

np.logical_and(a, b)

valor verdadero de x1 AND x2 (con arrays con valores booleanos o arrays con valores -1, 0, y 1))

np.logical_or(a, b)

evalúa el valor verdadero de x1 OR x2

np.logical_xor(a, b)

evalúa el valor verdadero de x1 XOR x2

np.logical_not(a)

aplica el operador lógico NOT a los elementos

np.maximum(a, b)

evalúa el valor máximo de las estructuras de entrada comparando elemento a elemento

np.minimum(a, b)

evalúa el valor mínimo de las estructuras de entrada

Unión de arrays

np.concatenate((a, b), axis = 1)

Unión de arrays. Eje por defecto es 0.

np.hstack((a, b))

concatena arrays horizontalmente (uno al lado del otro)

np.vstack((a, b))

realiza la concatenación vertical (uno encima del otro)