GOBIERNO DE
EL SALVADOR

SECRETARÍA DE
INNOVACIÓN

CIENCIA DE DATOS

Agenda
Sesión 13/18

SciPy

Parte 2

- Datos Espaciales
- Trabajar con matrices de Matlab
- Interpolación
- Prueba de Significación Estadística

Datos Espaciales

Trabajar con datos espaciales

Los datos espaciales se refieren a los datos que se representan en un espacio geométrico.

Por ejemplo, puntos en un sistema de coordenadas.

Nos ocupamos de problemas de datos espaciales en muchas tareas.

Por ejemplo, encontrar si un punto está dentro de un límite o no.

SciPy nos proporciona el módulo scipy.spatial, que tiene funciones para trabajar con datos espaciales.

Datos Espaciales

Triangulación

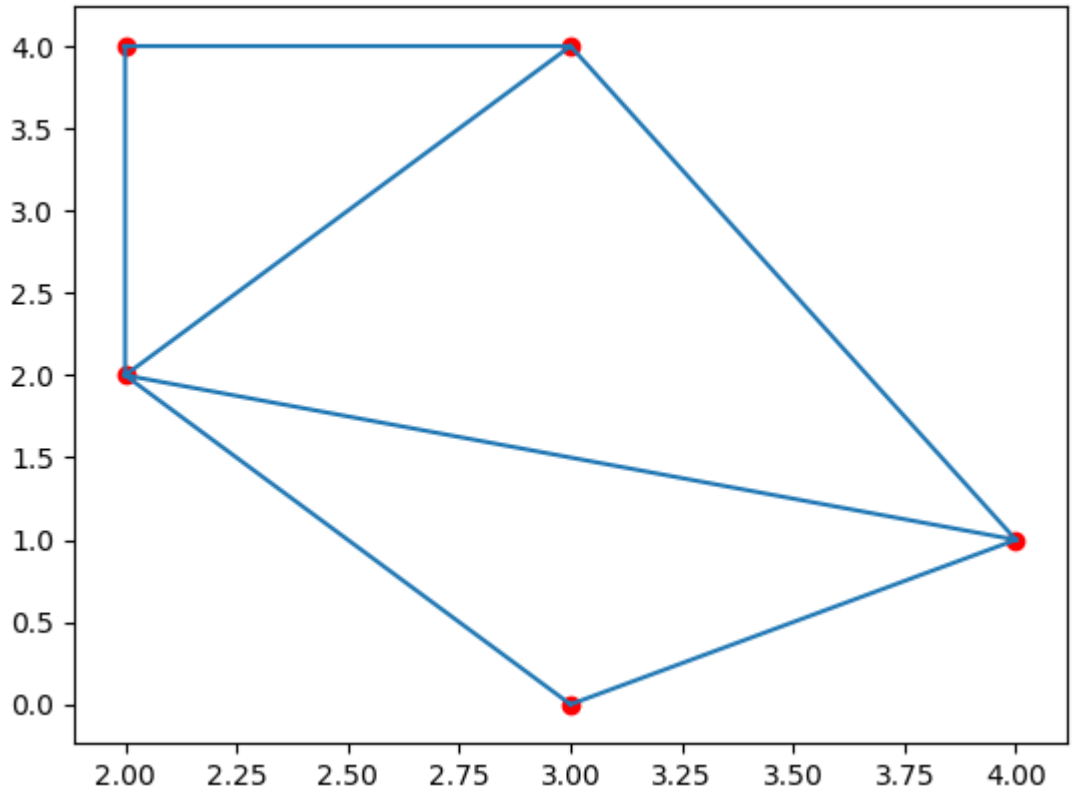
Una triangulación de un polígono consiste en dividir el polígono en varios triángulos con los que podemos calcular un área del polígono.

Una triangulación con puntos significa crear triángulos compuestos de superficie en los que todos los puntos dados están en al menos un vértice de cualquier triángulo de la superficie.

Un método para generar estas triangulaciones a través de puntos es la triangulación Delaunay().

Datos Espaciales

Nota: La propiedad simplices crea una generalización de la notación triangular.

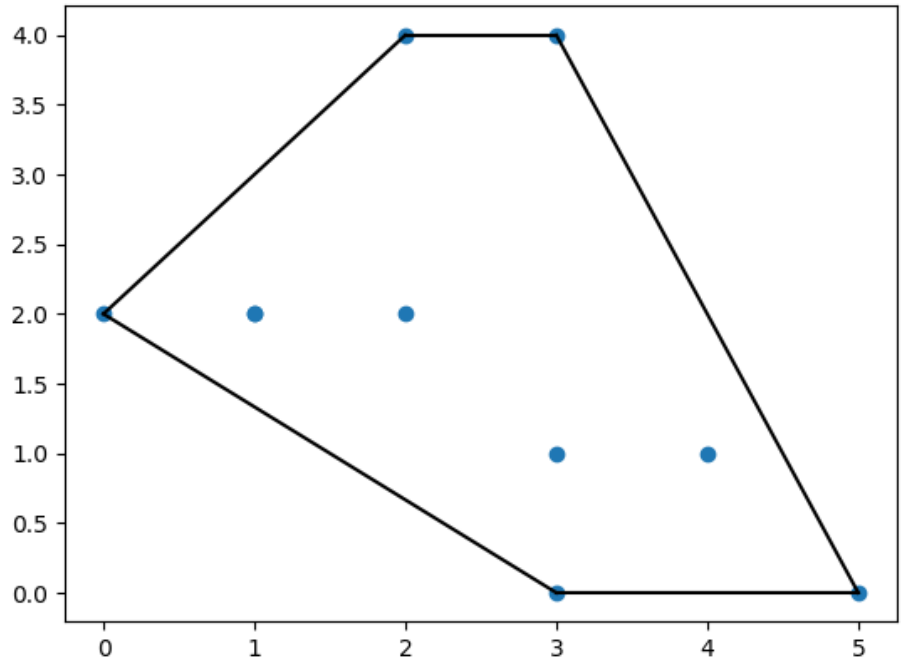


Datos Espaciales

Casco convexo

Un casco convexo es el polígono más pequeño que cubre todos los puntos dados.

Utilice el método ConvexHull() para crear un casco convexo.



Datos Espaciales

KDTrees

KDTrees son una estructura de datos optimizada para consultas de vecinos más cercanos.

Por ejemplo, en un conjunto de puntos usando KDTrees podemos preguntar de manera eficiente qué puntos están más cerca de un determinado punto.

El método KDTree() devuelve un objeto KDTree.

El método query() devuelve la distancia al vecino más cercano y la ubicación de los vecinos.

Datos Espaciales

```
from scipy.spatial import KDTree

points = [(1, -1), (2, 3), (-2, 3), (2, -3)]

kdtree = KDTree(points)

res = kdtree.query((1, 1))

print(res)
```

Datos Espaciales

Matriz de distancia

Hay muchas métricas de distancia que se utilizan para encontrar varios tipos de distancias entre dos puntos en la ciencia de datos, la distensión euclidiana, la distensión del coseno, etc.

La distancia entre dos vectores no solo puede ser la longitud de la línea recta entre ellos, también puede ser el ángulo entre ellos desde el origen, o el número de pasos unitarios requeridos, etc.

Gran parte del rendimiento del algoritmo de aprendizaje automático depende en gran medida de las métricas de distancia. Por ejemplo, "K vecinos más cercanos" o "K medias", etc.

Datos Espaciales

Distancia euclidiana

Calcula la distancia euclidiana entre puntos dados.

```
from scipy.spatial.distance import euclidean

p1 = (1, 0)
p2 = (10, 2)

res = euclidean(p1, p2)

print(res)
```

Coche unidireccional Cityblock

Es la distancia calculada usando 4 grados de movimiento.

Por ejemplo, solo podemos movernos: arriba, abajo, derecha o izquierda, no en diagonal.

```
from scipy.spatial.distance import cityblock

p1 = (1, 0)
p2 = (10, 2)
res = cityblock(p1, p2)
print(res)
```

Distancia coseno

Es el valor del ángulo coseno entre los dos puntos A y B.

```
from scipy.spatial.distance import cosine

p1 = (1, 0)
p2 = (10, 2)
res = cosine(p1, p2)
print(res)
```

Datos Espaciales

Distancia de Hamming

Es la proporción de bits en la que dos bits son diferentes.

Es una forma de medir la distancia para secuencias binarias.

```
from scipy.spatial.distance import hamming

p1 = (True, False, True)
p2 = (False, True, True)
res = hamming(p1, p2)
print(res)
```

Trabajar con matrices de Matlab

Trabajar con matrices de Matlab

Sabemos que NumPy nos proporciona métodos para conservar los datos en formatos legibles para Python. Pero SciPy también nos proporciona interoperabilidad con Matlab.

SciPy nos proporciona el módulo scipy.io, que tiene funciones para trabajar con matrices Matlab.

Trabajar con matrices de Matlab

Exportación de datos en formato Matlab

La función savemat() nos permite exportar datos en formato Matlab.

El método toma los siguientes parámetros:

nombre de archivo

el nombre del archivo para guardar los datos.

mdict

un diccionario que contiene los datos.

do_compression

un valor booleano que especifica si se comprime el resultado o no. Falso predeterminado.

Trabajar con matrices de Matlab

Exporte la siguiente matriz como nombre de variable "vec" a un archivo mat:

```
from scipy import io
import numpy as np
arr = np.arange(10)
io.savemat('arr.mat', {"vec": arr})
```

#Se guarda un nombre de archivo "arr.mat" en su computadora.

Trabajar con matrices de Matlab

Importar datos desde formato Matlab

La función loadmat() nos permite importar datos de un archivo Matlab.

La función toma un parámetro requerido:

nombre de archivo : el nombre de archivo de los datos guardados.

Devolverá una matriz estructurada cuyas claves son los nombres de las variables y los valores correspondientes son los valores de las variables.

Trabajar con matrices de Matlab

```
from scipy import io
import numpy as np
arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9,])

# Export:
io.savemat('arr.mat', {"vec": arr})
# Import:
mydata = io.loadmat('arr.mat')

print(mydata)

# Use el nombre de variable "vec" para mostrar solo la matriz de los datos de matlab
print(mydata['vec'])
```

Trabajar con matrices de Matlab

```
from scipy import io
import numpy as np
arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9,])

# Export:
io.savemat('arr.mat', {"vec": arr})
# Import:
mydata = io.loadmat('arr.mat')

print(mydata)

# Use el nombre de variable "vec" para mostrar solo la matriz de los datos de matlab
print(mydata['vec'])
```

Interpolación

¿Qué es la interpolación?

La interpolación es un método para generar puntos entre puntos dados.

Por ejemplo:

para los puntos 1 y 2, podemos interpolar y encontrar los puntos 1.33 y 1.66.

La interpolación tiene muchos usos, en Machine Learning a menudo tratamos con datos faltantes en un conjunto de datos, la interpolación se usa a menudo para sustituir esos valores.

Este método de llenar los valores se llama imputación .

Aparte de la imputación, la interpolación se usa a menudo cuando necesitamos suavizar los puntos discretos en un conjunto de datos.

Interpolación

¿Cómo implementarlo en SciPy?

SciPy nos proporciona un módulo llamado scipy.interpolate que tiene muchas funciones para lidiar con la interpolación

Interpolación 1D

La función interp1d() se utiliza para interpolar una distribución con 1 variable.

Toma “x” y “y” apunta y devuelve una función invocable que se puede llamar con new “x” y devuelve correspondiente “y”.

Interpolación

Para “x” y “y” dados, interpolar valores de 2.1, 2.2 ... a 2.9:

```
from scipy.interpolate import interp1d
import numpy as np
xs = np.arange(10)
ys = 2*xs + 1

interp_func  = interp1d(xs, ys)
newarr = interp_func(np.arange(2.1, 3, 0.1))
print(newarr)

#Nota: las nuevas x deben estar en el mismo rango que las antiguas x,
#lo que significa que no podemos llamar interp_func() con valores
#superiores a 10 o inferiores a 0.
```

Interpolación

Interpolación de splines

En la interpolación 1D, los puntos se ajustan para una sola curva, mientras que en la interpolación Spline los puntos se ajustan a una función por partes definida con polinomios llamados splines.

La función UnivariateSpline() toma xs y ys y produce una función invocable que se puede llamar con new xs.

Función por partes:

una función que tiene una definición diferente para diferentes rangos.

Interpolación

Encuentre la interpolación spline univariante para 2.1, 2.2 ... 2.9 para los siguientes puntos no lineales:

```
from scipy.interpolate import UnivariateSpline
import numpy as np

xs = np.arange(10)
ys = xs**2 + np.sin(xs) + 1

interp_func  = UnivariateSpline(xs, ys)
newarr = interp_func(np.arange(2.1, 3, 0.1))

print(newarr)
```

Prueba de Significación Estadística

En estadística, significancia estadística significa que el resultado que se produjo tiene una razón detrás, no se produjo al azar o por casualidad.

SciPy nos proporciona un módulo llamado scipy.stats, que tiene funciones para realizar pruebas de significación estadística.

A continuación, se muestran algunas técnicas y palabras clave que son importantes al realizar dichas pruebas:

Hipótesis en estadística

La hipótesis es una suposición sobre un parámetro en la población.

Prueba de Significación Estadística

Hipótesis nula

Asume que la observación no es estadísticamente significativa.

Hipótesis alternativa

Supone que las observaciones se deben a alguna razón.

Su alternativa a la hipótesis nula.

Ejemplo:
Para una evaluación de un estudiante tomaríamos:
"el estudiante es peor que el promedio" - como hipótesis nula, y:
"el estudiante es mejor que el promedio" - como hipótesis alternativa.

Prueba de Significación Estadística

Prueba de una cola

Cuando nuestra hipótesis está probando solo un lado del valor, se llama "prueba de una cola".

Ejemplo:

Para la hipótesis nula:

"la media es igual a k", podemos tener una hipótesis alternativa:

"la media es menor que k", o:

"la media es mayor que k"

Prueba de Significación Estadística

Prueba de dos colas

Cuando nuestra hipótesis está probando ambos lados de los valores.

Ejemplo:

Para la hipótesis nula:

"la media es igual a k", podemos tener una hipótesis alternativa:

"la media no es igual a k"

En este caso, la media es menor o mayor que k, y deben comprobarse ambos lados.

Prueba de Significación Estadística

Valor alfa

El valor alfa es el nivel de significancia.

Ejemplo:

Qué tan cerca de los extremos deben estar los datos para que se rechace una hipótesis nula. Por lo general, se toma como 0.01, 0.05 o 0.1.

Valor p

El valor P indica qué tan cerca del extremo están los datos.

El valor de p y los valores de alfa se comparan para establecer la significancia estadística.

Si p valor <= alfa, rechazamos la hipótesis nula y decimos que los datos son estadísticamente significativos. de lo contrario, aceptamos la hipótesis nula.

Prueba de Significación Estadística

Prueba T

Las pruebas T se utilizan para determinar si existe una deferencia significativa entre las medias de dos variables. y nos avisa si pertenecen a la misma distribución.

Es una prueba de dos colas.

La función ttest_ind() toma dos muestras del mismo tamaño y produce una tupla de estadística t y valor p.

Prueba de Significación Estadística

Encuentre si los valores dados v1 y v2 son de la misma distribución:

```
import numpy as np
from scipy.stats import ttest_ind

v1 = np.random.normal(size=100)
v2 = np.random.normal(size=100)

res = ttest_ind(v1, v2)

print(res)
```

Prueba de Significación Estadística

Encuentre si los valores dados v1 y v2 son de la misma distribución:

Si desea devolver solo el valor p, use la propiedad pvalue:

res = ttest_ind, v2).pvalue

print(res)

Prueba de Significación Estadística

Prueba KS

La prueba KS se utiliza para verificar si los valores dados siguen una distribución.

La función toma el valor a probar y el CDF como dos parámetros.

Un CDF puede ser una cadena o una función invocable que devuelve la probabilidad.

Se puede utilizar como prueba de una o dos colas.

Por defecto tiene dos colas. Podemos pasar la alternativa de parámetro como una cadena de uno de dos lados, menor o mayor.

Prueba de Significación Estadística

Encuentre si el valor dado sigue la distribución normal:

```
import numpy as np
from scipy.stats import kstest

v = np.random.normal(size=100)

res = kstest(v, 'norm')

print(res)
```

Prueba de Significación Estadística

Descripción estadística de datos

Para ver un resumen de valores en una matriz, podemos usar la función describe().

Devuelve la siguiente descripción:

1. número de observaciones (nobs)
2. valores mínimo y máximo = minmax
3. significar
4. diferencia
5. oblicuidad
6. curtosis

Prueba de Significación Estadística

Mostrar descripción estadística de los valores en una matriz:

```
import numpy as np
from scipy.stats import describe

v = np.random.normal(size=100)
res = describe(v)

print(res)
```

Prueba de Significación Estadística

Pruebas de normalidad (asimetría y curtosis)

Las pruebas de normalidad se basan en la asimetría y la curtosis.

La función normaltest() devuelve el valor p para la hipótesis nula:

"x proviene de una distribución normal" .

Prueba de Significación Estadística

Oblicuidad: Una medida de simetría en los datos.

Para distribuciones normales es 0.
Si es negativo, significa que los datos están sesgados hacia la izquierda. Si es positivo, significa que los datos están sesgados a la derecha.

Curtosis: Una medida de si los datos son pesados o tienen una ligera cola en una distribución normal.

La curtosis positiva significa cola pesada.
La curtosis negativa significa con una cola ligera.

Prueba de Significación Estadística

Encuentre asimetría y curtosis de valores en una matriz:

```
import numpy as np
from scipy.stats import skew, kurtosis
v = np.random.normal(size=100)
print(skew(v))
print(kurtosis(v))
```

Encuentre si los datos provienen de una distribución normal:

```
import numpy as np
from scipy.stats import normaltest
v = np.random.normal(size=100)
print(normaltest(v))
```

RESUMEN DE CLASE

GOBIERNO DE
EL SALVADOR

SECRETARÍA DE
INNOVACIÓN