**距离**

**欧氏距离**

from scipy.spatial.distance import Euclidean

distance=euclidean(point1, point2)

**曼哈顿距离**

from scipy.spatial.distance import cityblock

distance= cityblock(point1, point2)

**Cosine距离**

from scipy.spatial.distance import cosine

distance= cosine(point1, point2)

**一维插值**

from scipy import interpolate

interpolate\_func= interpolate.interp1d (x,y)

y\_interpolated=interpolate\_func(x\_new)

#基于x,y生成插值函数，再基于待插值位置x\_new计算y\_interpolated

#x\_new需处于用于生成插值函数的x的范围中

**稀疏矩阵**

**创建稀疏矩阵**

from scipy.sparse import csr\_matrix

res=csr\_matrix(matrixdata)

print(res)

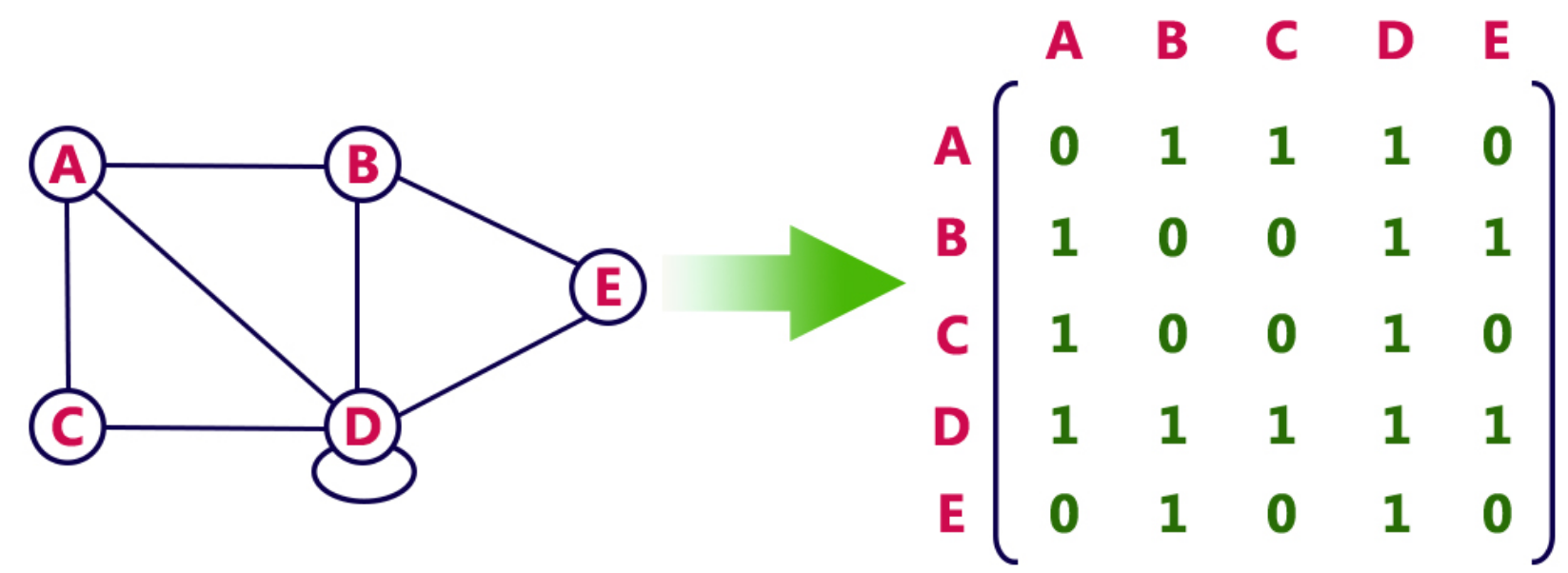
#记录非零元素行列位置、值

**非零元素计数**

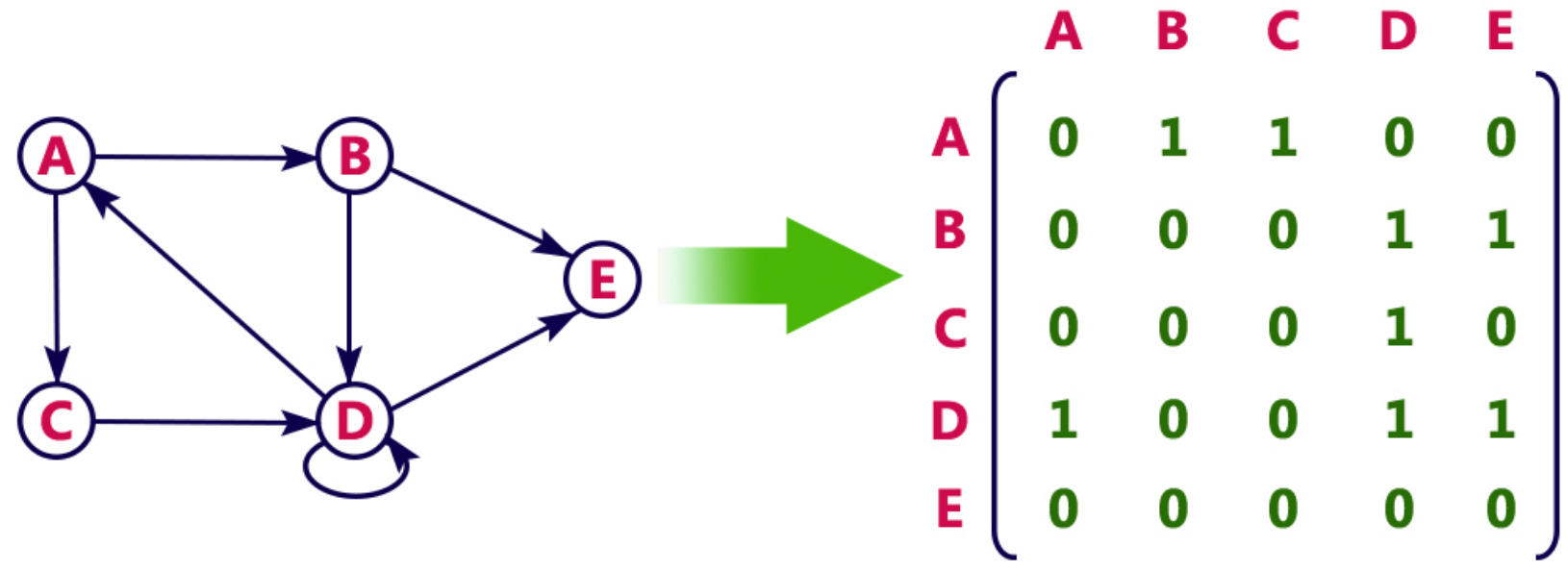
res.count\_nonzero()

**图**

**无向图邻接矩阵**



**有向图邻接矩阵**



#有向图邻接矩阵列名为接收点，行名为发出点

**统计描述**

from scipy.stats import describe

res = describe(data)

**最优化**

用于求解在条件内使得目标函数fun取得最小值的参数list

from scipy.optimize import minimize

res=minimize(fun,x0=参数list起始估计值,bounds,constraints)

res.x #所估计参数

e.g.

fun=lambda x: x[0]+2x[1]-3x[2]

#常用于求最优参数，将fun设置为loss函数

e.g.

def loss(X\_train,y\_train, model\_params):

return some\_metric(y\_train, model(X\_train, model\_params))

**假设检验**

**Anderson-Darling检验**

import scipy.stats as stats

stats.anderson(data\_norm, dist='norm')

[**https://blog.csdn.net/qq\_20207459/article/details/102863982**](https://blog.csdn.net/qq_20207459/article/details/102863982)

**KS test**

from scipy.stats import kstest

res = kstest(data, 'norm', alternative={'two-sided', 'less', 'greater'})

**直接正态检验**

from scipy.stats import normaltest

print(normaltest(data, nan\_policy={'propagate', 'raise', 'omit'}))

**R2拟合优度**

RSS=sum(pow(y\_pred-y\_true,2))

ESS=sum(pow(y\_pred-np.mean(y\_true),2))

TSS=sum(pow(y\_true-np.mean(y\_true),2))

r=np.sqrt(ESS/TSS)

print('r value:',r)

**独立双样本t检验**

from scipy import stats

res=stats.ttest\_ind(Group1,Group2, equal\_var=True, nan\_policy={'propagate', 'raise', 'omit'})

print(res)

#euqal\_var控制默认两个群体方差是否相同

**配对t检验**

from scipy import stats

res=stats.ttest\_rel(Group1,Group2, nan\_policy={'propagate', 'raise', 'omit'})

print(res)

**获取p值**

res.pvalue

**单因素ANOVA**

from scipy import stats

res=stats.f\_oneway(level1data,level2data,…)

print(res)

**F检验**

from scipy.stats import f\_oneway

scipy.stats.f\_oneway(data1,data2)

**卡方检验**

from scipy.stats import chi2\_contingency

res=chi2\_contingency(contingency\_table)

print(res)

#return (chi2,p,自由度,理论频率)