*VII-3-2.階層的クラスター分析*

リストVII-3-2-i.分析準備・主成分分析

|  |
| --- |
| *#[A]必要なライブラリーの読み込み*  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  %matplotlib inline  from sklearn.model\_selection import train\_test\_split  from scipy.cluster.hierarchy import linkage,dendrogram,fcluster  *#[B]データの読み込み*  df =pd.read\_csv("sample10.csv")  xn,D=df.shape  D1=D-1  *#データフレームをつくる*  dfX=pd.DataFrame(df)  X=df.values  X=np.delete(X,0,1)  *#列名を付ける*  for i in range (D):  dfX=dfX.rename(columns={i:"X"+str(i+1)})  print (dfX)  D1=D-1  *#[C]標準化後主成分分析を実行*  import urllib.request  import matplotlib.pyplot as plt  import sklearn  *#機械学習のライブラリ*  from sklearn.decomposition import PCA #主成分分析  from sklearn.preprocessing import StandardScaler *#標準化*  from IPython.display import display  *#標準化*  std\_sc = StandardScaler()  std\_sc.fit(X)  std\_data = std\_sc.transform(X)  std\_data\_df = pd.DataFrame(std\_data)  display(std\_data\_df)  *#主成分分析の実行*  pca = PCA()  pca.fit(std\_data\_df)  *# データを主成分空間に写像*  pca\_cor = pca.transform(std\_data\_df)  print(pca.get\_covariance()) *# 分散共分散行列*  *# 固有ベクトルのマトリックス表示*  eig\_vec = pd.DataFrame(pca.components\_.T, \  columns = ["PC{}".format(x + 1) for x in range(len(std\_data\_df.columns))])  display(eig\_vec)  *# 固有値*  eig = pd.DataFrame(pca.explained\_variance\_, index=["PC{}".format(x + 1) for x in range(len(std\_data\_df.columns))], columns=['固有値']).T  display(eig)  *# Rによるソースコードだと、固有値（分散）ではなく標準偏差を求めている。*  *# 主成分の標準偏差*  dv = np.sqrt(eig)  dv = dv.rename(index = {'固有値':'主成分の標準偏差'})  display(dv)  *# 寄与率*  ev = pd.DataFrame(pca.explained\_variance\_ratio\_, index=["PC{}".format(x + 1) for x in range(len(std\_data\_df.columns))], columns=['寄与率']).T  display(ev)  *# 累積寄与率*  t\_ev = pd.DataFrame(pca.explained\_variance\_ratio\_.cumsum(), index=["PC{}".format(x + 1) for x in range(len(std\_data\_df.columns))], columns=['累積寄与率']).T  display(t\_ev)  *# 主成分得点*  print('主成分得点')  cor = pd.DataFrame(pca\_cor, columns=["PC{}".format(x + 1) for x in range(len(std\_data\_df.columns))])  display(cor)  PC=cor.values  dfS=pd.concat([dfX,cor],axis=1)  S=dfS.values |

リストVII-3-2-ii.主成分の数を決定し、データをシャッフルしてとレーニングデータとテストデータに分けて保存

|  |
| --- |
| *#[A]主成分寄与率・累積寄与率から主成分の数を決定する*  P=2  *#[B]データをシャッフルしトレーニングデータとテストデータに分割して保存*  TrainingRatio=0.5  S\_train, S\_test = train\_test\_split(S, train\_size=TrainingRatio, random\_state=1)  n,cul=S\_train.shape  T\_train=np.zeros((n,1))  X\_train=np.zeros((n,D1))  PC\_train=np.zeros((n,P))  for i in range(1):  T\_train[:,i]=S\_train[:,i]  for i in range(D1):  X\_train[:,i]=S\_train[:,1+i]  for i in range(P):  PC\_train[:,i]=S\_train[:,1+D1+i]  n,col=S\_test.shape  T\_test=np.zeros((n,1))  X\_test=np.zeros((n,D1))  PC\_test=np.zeros((n,P))  for i in range(1):  T\_test[:,i]=S\_test[:,i]  for i in range(D1):  X\_test[:,i]=S\_test[:,1+i]  for i in range(P):  PC\_test[:,i]=S\_test[:,1+D1+i] |

リストVII-3-2-iii.データ分布の確認（クラスの識別ナシ）

|  |
| --- |
| *#データ分布の確認*  *#[A]散布図の描画法を定義する*  def show\_data(x): 　plt.plot(x[:,x0],x[:,y0],linestyle='none',marker='o',markeredgecolor='black',color="white",alpha=0.8)  plt.grid(True)  def show\_data1(x,t):  col=["b","r","g","y","w","c","m","k"]  for c in range (C): plt.plot(x[t[:,0]==c+1,x0],x[t[:,0]==c+1,y0],linestyle='none',marker='o',markeredgecolor='black',color=col[c],alpha=0.8)  plt.grid(True)#元データの分布  *#[B]元データの散布図*  *#変数の選択*  x=1  y=2  x\_range=[-2,2] *#項目1の範囲*  y\_range=[-2,2] *#項目2の範囲*  x0=x-1  y0=y-1  #実行  plt.figure(1,figsize=(8,3.7))  plt.subplot(1,2,1)  show\_data(X\_train)  plt.xlim(x\_range)  plt.ylim(y\_range)  plt.xlabel("X"+str(x))  plt.ylabel("X"+str(y))  plt.title('Training data')  plt.subplot(1,2,2)  show\_data(X\_test)  plt.xlim(x\_range)  plt.ylim(y\_range)  plt.xlabel("X"+str(x))  plt.ylabel("X"+str(y))  plt.title('Test data')  plt.show()  *#[C]主成分得点のデータ分布の確認*  *#主成分の選択*  x=1  y=2  x\_range=[-3,3] #項目1の範囲  y\_range=[-3,3] #項目2の範囲  x0=x-1  y0=y-1  #実行  plt.figure(1,figsize=(8,3.7))  plt.subplot(1,2,1)  show\_data(PC\_train)  plt.xlim(x\_range)  plt.ylim(y\_range)  plt.xlabel("PC"+str(x))  plt.ylabel("PC"+str(y))  plt.title('PC\_Training data')  plt.subplot(1,2,2)  show\_data(PC\_test)  plt.xlim(x\_range)  plt.ylim(y\_range)  plt.xlabel("PC"+str(x))  plt.ylabel("PC"+str(y))  plt.title('PC\_Test data')  plt.show() |

リストVII-3-2-iv.データ分布の確認（クラスの識別つき）

|  |  |
| --- | --- |
| *#データ分布の確認(クラス識別)*  *#[A]元データの分布*  C=5 *#クラスの数*  *#変数の選択*  x=1  y=2  x0=x-1  y0=y-1  x\_range=[-2,2] *#項目1の範囲*  y\_range=[-2,2] *#項目2の範囲*  plt.figure(1,figsize=(8,3.7))  plt.subplot(1,2,1)  show\_data1(X\_train,T\_train)  plt.xlim(x\_range)  plt.ylim(y\_range)  plt.xlabel("X"+str(x))  plt.ylabel("X"+str(y))  plt.title('Training data')  plt.subplot(1,2,2)  show\_data1(X\_test,T\_test)  plt.xlim(x\_range)  plt.ylim(y\_range)  plt.xlabel("X"+str(x))  plt.ylabel("X"+str(y))  plt.title('Test data')  plt.show()  *#[B]主成分得点のデータ分布の確認*  *#主成分の選択*  x=1  y=2  x0=x-1  y0=y-1  x\_range=[-3,3] *#項目1の範囲*  y\_range=[-3,3] *#項目2の範囲*  plt.figure(1,figsize=(8,3.7))  plt.subplot(1,2,1)  show\_data1(PC\_train,T\_train)  plt.xlim(x\_range)  plt.ylim(y\_range)  plt.xlabel("PC"+str(x))  plt.ylabel("PC"+str(y))  plt.title('PC\_Training data')  plt.subplot(1,2,2)  show\_data1(PC\_test,T\_test)  plt.xlim(x\_range)  plt.ylim(y\_range)  plt.xlabel("PC"+str(x))  plt.ylabel("PC"+str(y))  plt.title('PC\_Test data')  plt.show() |  |

リストVII-3-2-v. ユークリッド距離を非類似度として単連結法でクラスター分析

|  |
| --- |
| *#ユークリッド距離を非類似度として単連結法でクラスター分析*  z1 = linkage(X\_train, metric='euclidean', method="single")  z2 = linkage(X\_test, metric='euclidean', method="single")  *# 結果を可視化*  plt.figure(1,figsize=(8,3.7))  plt.subplot(1,2,1)  dendrogram(z1)  plt.title("Training data.euclid-single")  plt.subplot(1,2,2)  dendrogram(z2)  plt.title("Test data.euclid-single")  plt.show() |

リストVII-3-2-vi. 上記のクラスター分析の結果を散布図で表す。

|  |
| --- |
| *#クラス分けの結果を散布図で表示*  *#[A]クラスの数を決める*  C=4  *#変数の選択*  x=1  y=2  x0=x-1  y0=y-1  x\_range=[-2,2] *#項目1の範囲*  y\_range=[-2,2] *#項目2の範囲*  *#[B]training dataの分布*  clusters = fcluster(z1, t=C, criterion='maxclust')*#用いるデンドログラムを指定*  n,nn=X\_train.shape  for i in range(n):  T\_train[i,0]=clusters[i]  plt.figure(1,figsize=(8,3.7))  plt.subplot(1,2,1)  show\_data1(X\_train,T\_train)  plt.xlim(x\_range)  plt.ylim(y\_range)  plt.xlabel("X"+str(x))  plt.ylabel("X"+str(y))  plt.title('Training data')  *#[C]test dataの分布*  clusters = fcluster(z2, t=C, criterion='maxclust')　*#用いるデンドログラムを指定*  n,nn=X\_test.shape  for i in range(n):  T\_test[i,0]=clusters[i]  plt.subplot(1,2,2)  show\_data1(X\_test,T\_test)  plt.xlim(x\_range)  plt.ylim(y\_range)  plt.xlabel("X"+str(x))  plt.ylabel("X"+str(y))  plt.title('Test data')  plt.show() |

リストVII-3-2-vii. デンドログラムの作成　（ユークリッド距離・完全連結法）

|  |
| --- |
| *#＊＊＊＊＊を非類似度として＊＊＊＊＊法でクラスター分析*  z3 = linkage(X\_train, metric='euclidean', method="complete")　*#[A]*  z4 = linkage(X\_test, metric='euclidean', method="complete") *#[B]*  *# 結果を可視化*  plt.figure(1,figsize=(8,3.7))  plt.subplot(1,2,1)  dendrogram(z3)  plt.title("Training data.euclid-complete")  plt.subplot(1,2,2)  dendrogram(z4)  plt.title("Test data.euclid-complete")  plt.show() |

リストVII-3-2-viii.(データ分布とクラス分け、ユークリッド距離・完全連結法）

|  |
| --- |
| *#クラス分けの結果を散布図で表示*  *#[A]クラスの数を決める*  *C=5*  *#変数の選択*  *x=1*  *y=2*  *x0=x-1*  *y0=y-1*  *x\_range=[-2,2] #項目1の範囲*  *y\_range=[-2,2] #項目2の範囲*  *#[B]training dataの分布*  *clusters = fcluster(z3, t=C, criterion='maxclust')#クラスの数を決める*  *n,nn=X\_train.shape*  *for i in range(n):*  *T\_train[i,0]=clusters[i]*  *plt.figure(1,figsize=(8,3.7))*  *plt.subplot(1,2,1)*  *show\_data1(X\_train,T\_train)*  *plt.xlim(x\_range)*  *plt.ylim(y\_range)*  *plt.xlabel("X"+str(x))*  *plt.ylabel("X"+str(y))*  *plt.title('Training data')*  *#[C]test dataの分布*  *clusters = fcluster(z4, t=C, criterion='maxclust')*  *n,nn=X\_test.shape*  *for i in range(n):*  *T\_test[i,0]=clusters[i]*  *plt.subplot(1,2,2)*  *show\_data1(X\_test,T\_test)*  *plt.xlim(x\_range)*  *plt.ylim(y\_range)*  *plt.xlabel("X"+str(x))*  *plt.ylabel("X"+str(y))*  *plt.title('Test data')*  *plt.show()* |

連結法（method）と距離(metric)のコード

表72.scipy.clusterにおける連結法(引数method)のコード

|  |  |
| --- | --- |
| 連結法 | コード（引数method= |
| 単連結法（simple linkage method） | single |
| 完全連結法(complete linkage method) | complet |
| 群平均法(group average method) | average |
| 重み付き平均法（weighted average method） | weighted |
| 重心法(centroid method) | centroid |
| ウォード法(Ward’s method) | ward |

表73.scipy.clusterにおける距離(非類似度)(引数metric)のコード

|  |  |
| --- | --- |
| 距離(非類似度) | コード（metric=） |
| ユークリッド距離 | euclidian |
| ユークリッド距離の２乗 | sqeuclidian |
| 標準化されたユークリッド距離 | seuclidian |
| マハラノビス距離 | mahalanobis |
| ミンコフスキー距離 | mincowski |
| マンハッタン距離 | cityblock |
| キャンベラ距離 | canberra |
| 1-Pearsonの相関係数 | correlation |
| コサイン非類似度 | cosine |
| ジャカード非類似度 | jaccard |
| ダイス非類似度 | dice |