*VII-3-4-1. 混合ガウスモデル(scikit.learn.mixutureを使わない)*

VII-3-4-1-i データの読み込み、主成分分析、（VII-3-2-i.と同じ）

|  |
| --- |
| #[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  import decimal  decimal.getcontext().prec=6  #[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)  PCC=cor.values  dfS=pd.concat([dfX,cor],axis=1)  S=dfS.values |

VII-3-4-1-ii. 主成分数の決定

|  |
| --- |
| #主成分の数を決定する  P=2 #主成分の数を入力  N,D=PCC.shape  PC=np.zeros((N,P))  for n in range(N):  for p in range (P):  PC[n,p]=PCC[n,p] |

VII-3-4-1.iii.分析するデータを決定し、初期値を与える。

|  |
| --- |
| X=X  N,D=X.shape  X\_range0=[-2,2]  X\_range1=[-2,2]  #マーカーの色決定  x\_col=np.array([[0,0,0.95],[0.95,0,0],[0,0.95,0],[0.95,0.95,0],[1,1,1],[0,0.95,0.95],[0,0,0],[0.95,0,0.95]])  #初期条件  Pi0=np.array([0.2,0.2,0.2,0.2,0.2])  Pi=Pi0  Mu0=np.array([[1,1],[-1,1],[-1,-1],[1,-1],[0,0]])  Mu=Mu0  Sigma0=np.array([[[1,0],[0,1]],[[1,0],[0,1]],[[1,0],[0,1]],[[1,0],[0,1]],[[1,0],[0,1]]])  Sigma=Sigma0  N=X.shape[0]  K=len(Pi)  Gamma0=np.c\_[np.ones((N,1)),np.zeros((N,K-1))]  Gamma=Gamma0 |

リストVII-3-4-iv.関数の定義

|  |
| --- |
| #関数の定義  #ガウス関数の定義  def gauss(x,mu,sigma):  N,D=x.shape  c1=-(D/2)\*np.log(2\*np.pi)  det\_sigma=np.linalg.det(sigma)  c2=-(1/2)\*np.log(det\_sigma)  inv\_sigma=np.linalg.inv(sigma)  c3=x-mu  c4=np.dot(c3,inv\_sigma)  c5=np.zeros(N)  for d in range(D):  c5=c5+c4[:,d]\*c3[:,d]  c5=-c5/2  p=c1+c2+c5  p=np.exp(p)  return p  #混合ガウスモデル  def mixgauss(x,pi,mu,sigma):  N,D=x.shape  K=len(pi)  p=np.zeros(N)  for k in range(K):  p=p+pi[k]\*gauss(x,mu[k,:],sigma[k,:,:])  return p  #e step (gammaの更新)  def e\_step\_mixgauss(x,pi,mu,sigma):  N,D=x.shape  K=len(pi)  y=np.zeros((N,K))  for k in range(K):  y[:,k]=gauss(x,mu[k,:],sigma[k,:,:])#クラスｋの分布でXが得られる確率（分子）  gamma=np.zeros((N,K))  for n in range(N):  wk=np.zeros(K)  for k in range(K):  wk[k]=pi[k]\*y[n,k]  gamma[n,:]=wk/np.sum(wk)  return gamma  #m step (Pi,Mu,Sigmaの更新)  def m\_step\_mixgauss(x,gamma):  N,D=x.shape  N,K=gamma.shape  #piを計算  pi=np.sum(gamma,axis=0)/N  #muを計算  mu=np.zeros((K,D))  for k in range(K):  for d in range(D):  mu[k,d]=np.dot(gamma[:,k],x[:,d])/np.sum(gamma[:,k])  #sigmaを計算  sigma=np.zeros((K,D,D))  for k in range(K):  for n in range(N):  wk=x-mu[k,:]  wk=wk[n,:,np.newaxis]  sigma[k,:,:]=sigma[k,:,:]+gamma[n,k]\*np.dot(wk,wk.T)  sigma[k,:,:]=sigma[k,:,:]/np.sum(gamma[:,k])  return pi,mu,sigma  #EMアルゴリズム  def em\_alg(max\_it,err,pi,mu,sigma):  it=0  for it in range(0,max\_it):  gamma=e\_step\_mixgauss(X,pi,mu,sigma)  err[it]=nlh\_mixgauss(X,pi,mu,sigma)  pi,mu,sigma1=m\_step\_mixgauss(X,gamma)  return err,pi,mu,sigma,gamma  #誤差関数の定義  def nlh\_mixgauss(x,pi,mu,sigma):  N,D=x.shape  K=len(pi)  y=np.zeros((N,K))  for k in range(K):  y[:,k]=gauss(x,mu[k,:],sigma[k,:,:])  lh=0  for n in range(N):  wk=0  for k in range(K):  wk=wk+pi[k]\*y[n,k]  lh=lh+np.log(wk)  return -lh  #尤度・確率計算  #個々入力データの尤度  def likelihood(xx,mu,pi,sigma):  N,D=xx.shape  ppk1=np.zeros((N,K))  ppl1=np.zeros((N,K))  g1=np.zeros((N,K))  S1=np.zeros((N))  for k in range(K):  g1[:,k]=gauss(xx,mu[k,:],sigma[k,:,:] )  ppk1[:,k]=pi[k]\*g1[:,k]  for n in range(N):  for k in range(K):  S1[n]=S1[n]+ppk1[n,k]  for k in range(K):  ppl1[n,k]=g1[n,k]\*ppk1[n,k]/S1[n]  ratio1=np.zeros((N,K))  for n in range(N):  SS1=0  for k in range(K):  SS1=SS1+ppl1[n,k]  for k in range(K):  ratio1[n,k]=ppl1[n,k]/SS1  return g1,ppk1,ppl1,ratio1  #データの図示  #混合ガウス等高線表示  import matplotlib.pyplot as plt  from mpl\_toolkits.mplot3d import axes3d  %matplotlib inline  def show\_contour\_mixgauss(pi,mu,sigma):  xn=40 #解像度  x0=np.linspace(X\_range0[0],X\_range0[1],xn)  x1=np.linspace(X\_range1[0],X\_range1[1],xn)  xx0,xx1=np.meshgrid(x0,x1)  A=xx0.reshape(xn\*xn,1)  B=xx1.reshape(xn\*xn,1)  x=np.c\_[B,A]  f=mixgauss(x,pi,mu,sigma)  f=f.reshape(xn,xn)  f=f.T  plt.contour(x0,x1,f,10,color="grey")  #混合ガウス3D表示  def show3d\_mixgauss(ax,pi,mu,sigma):  xn=40 #解像度  x0=np.linspace(X\_range0[0],X\_range0[1],xn)  x1=np.linspace(X\_range1[0],X\_range1[1],xn)  xx0,xx1=np.meshgrid(x0,x1)  A=xx0.reshape(xn\*xn,1)  B=xx1.reshape(xn\*xn,1)  x= np.c\_[B,A]  f=mixgauss(x,pi,mu,sigma)  f=f.reshape(xn,xn)  f=f.T  ax.plot\_surface(xx0,xx1,f,rstride=2,cstride=2,alpha=0.3,color='blue',edgecolor='black')  #データ色分け等高線付き  def show\_mixgauss\_prm(x,gamma,pi,mu,sigma):  show\_contour\_mixgauss(pi,mu,sigma)  for n in range(N):  col=np.zeros(3)  for k in range(K):  col=col+gamma[n,k]\*x\_col[k]  plt.plot(x[n,0],x[n,1],'o',color=tuple(col),markeredgecolor='black',markersize=6,alpha=1)  for k in range(K):  plt.plot(mu[k,0],mu[k,1],marker='\*',markerfacecolor=tuple(x\_col[k]),markersize=15,markeredgecolor='k',markeredgewidth=1)  plt.grid(True)  #データ色分け等高なし、軸を選択  def show\_mixgauss\_prm2(x,gamma,pi,mu,sigma):  for n in range(N):  col=np.zeros(3)  for k in range(K):  col=col+gamma[n,k]\*x\_col[k,:]  plt.plot(x[n,d0],x[n,d1],'o',color=tuple(col),markeredgecolor='black',markersize=6,alpha=1)  for k in range(K):  plt.plot(mu[k,d0],mu[k,d1],marker='\*',markerfacecolor=tuple(x\_col[k]),markersize=15,markeredgecolor='k',markeredgewidth=1)  plt.grid(True)  def show\_mixgauss\_prm3(x,pi,mu,sigma,ratio):  show\_contour\_mixgauss(pi,mu,sigma)  for n in range(N):  col=np.zeros(3)  for k in range(K):  col=col+ratio[n,k]\*x\_col[k]  plt.plot(x[n,0],x[n,1],'o',color=tuple(col),markeredgecolor='black',markersize=6,alpha=1)  for k in range(K):  plt.plot(mu[k,0],mu[k,1],marker='\*',markerfacecolor=tuple(x\_col[k]),markersize=15,markeredgecolor='k',markeredgewidth=1)  plt.grid(True)  #データ色分け等高なし、軸を選択  def show\_mixgauss\_prm4(x,pi,mu,ratio):  for n in range(N):  col=np.zeros(3)  for k in range(K):  col=col+ratio[n,k]\*x\_col[k,:]  plt.plot(x[n,d0],x[n,d1],'o',color=tuple(col),markeredgecolor='black',markersize=6,alpha=1)  for k in range(K):  plt.plot(mu[k,d0],mu[k,d1],marker='\*',markerfacecolor=tuple(x\_col[k]),markersize=15,markeredgecolor='k',markeredgewidth=1)  plt.grid(True) |

VII-3-4-1-v.等高線図と3dグラフの動作確認

|  |
| --- |
| #混合ガウス関数（等高線と3d)  #等高線図と3d  Gamma2,Ppk1,Ppl1,Ratio1=likelihood(X,Mu,Pi,Sigma)  Fig=plt.figure(1,figsize=(8,3.5))  Fig.add\_subplot(1,2,1)  show\_mixgauss\_prm3(X,Pi,Mu,Sigma,Ratio1)  plt.grid(True)  Ax=Fig.add\_subplot(1,2,2,projection='3d')  show3d\_mixgauss(Ax,Pi, Mu,Sigma)  Ax.set\_xlabel('$x\_0$',fontsize=14)  Ax.set\_ylabel('$x\_1$',fontsize=14)  Ax.view\_init(40,-100)  plt.xlim(X\_range0)  plt.ylim(X\_range1)  plt.show() |

VII-3-4-1-vi.Eステップの確認とGammaの適正化

|  |
| --- |
| #e-stepの動作試験  Gamma=e\_step\_mixgauss(X,Pi,Mu,Sigma)  plt.figure(1,figsize=(4,4))  show\_mixgauss\_prm3(X,Pi,Mu,Sigma,Ratio1)  plt.show() |

VII-3-4-1-vii.動作試験( Mステップ)

|  |
| --- |
| #m-stepの動作試験  Pi,Mu,Sigma=m\_step\_mixgauss(X,Gamma)  Gamma2,Ppk1,Ppl1,Ratio1=likelihood(X,Mu,Pi,Sigma)  plt.figure(1,figsize=(4,4))  show\_mixgauss\_prm3(X,Pi,Mu,Sigma,Ratio1)  plt.show()  print(Mu) |

VII-3-4-1-viii. EMアルゴリズムの実施

|  |
| --- |
| #試行回数を決めてEMアルゴリズムを実施  max\_it=400  Err=np.zeros(max\_it)  Err1=Err  Pi1=Pi  Mu1=Mu  Sigma1=Sigma  Err2,Pi2,Mu2,Sigma2,Gamma2=em\_alg(max\_it,Err1,Pi1,Mu1,Sigma1)  plt.figure(2,figsize=(4,4))  plt.plot(np.arange(max\_it)+1,Err2,color='k',linestyle='-',marker='o')  plt.grid(True)  plt.show()  print(Mu2) |

VII-3-4-1-ix. 入力データの尤度

|  |
| --- |
| #個々入力データの尤度  xx=X  Gamma2,Ppk1,Ppl1,Ratio1=likelihood(xx,Mu2,Pi2,Sigma2)  print(Ppl1)  print(Ratio1) |

VII-3-4-1-x. 結果の図示

|  |
| --- |
| #結果を色分け図で表示する。  #表示する軸を選択する  #変数の選択  dim1=1  dim2=2  x\_range=[-2,2] #項目1の範囲  y\_range=[-2,2] #項目2の範囲  d0=dim1-1  d1=dim2-1  plt.figure(2,figsize=(4,4))  show\_mixgauss\_prm4(X,Pi2,Mu2,Ratio1)  plt.show() |

VII-3-4-1-xi. 混合ガウスモデルの再計算

|  |
| --- |
| #初期条件  Pi2=Pi0  Mu2=Mu1  Sigma2=Sigma0  N=X.shape[0]  K=len(Pi)  Gamma2=Gamma  plt.figure(1,figsize=(10,6.5))  max\_it=200  Err2=np.zeros((max\_it))  i\_subplot=1;  for it in range(0,max\_it):  Gamma2=e\_step\_mixgauss(X,Pi2,Mu2,Sigma2)  Err2[it]=nlh\_mixgauss(X,Pi2,Mu2,Sigma2)  Pi2,Mu2,Sigma2=m\_step\_mixgauss(X,Gamma2)  Gamma2,Ppk4,Ppl4,Ratio4=likelihood(xx,Mu2,Pi2,Sigma2)  if it<=0:  plt.subplot(2,3,i\_subplot)  show\_mixgauss\_prm3(X,Pi2,Mu2,Sigma2,Ratio4)  plt.title("{0:d}".format(it+1))  plt.xticks(range(X\_range0[0],X\_range0[1]),"")  plt.yticks(range(X\_range1[0],X\_range1[1]),"")  i\_subplot=i\_subplot+1  if it>3 and it<5:  plt.subplot(2,3,i\_subplot)  show\_mixgauss\_prm3(X,Pi2,Mu2,Sigma2,Ratio4)  plt.title("{0:d}".format(it+1))  plt.xticks(range(X\_range0[0],X\_range0[1]),"")  plt.yticks(range(X\_range1[0],X\_range1[1]),"")  i\_subplot=i\_subplot+1  if it>8 and it<10:  plt.subplot(2,3,i\_subplot)  show\_mixgauss\_prm3(X,Pi2,Mu2,Sigma2,Ratio4)  plt.title("{0:d}".format(it+1))  plt.xticks(range(X\_range0[0],X\_range0[1]),"")  plt.yticks(range(X\_range1[0],X\_range1[1]),"")  i\_subplot=i\_subplot+1  if it>48 and it<50:  plt.subplot(2,3,i\_subplot)  show\_mixgauss\_prm3(X,Pi2,Mu2,Sigma2,Ratio4)  plt.title("{0:d}".format(it+1))  plt.xticks(range(X\_range0[0],X\_range0[1]),"")  plt.yticks(range(X\_range1[0],X\_range1[1]),"")  i\_subplot=i\_subplot+1  if it>98 and it<100:  plt.subplot(2,3,i\_subplot)  show\_mixgauss\_prm3(X,Pi2,Mu2,Sigma2,Ratio4)  plt.title("{0:d}".format(it+1))  plt.xticks(range(X\_range0[0],X\_range0[1]),"")  plt.yticks(range(X\_range1[0],X\_range1[1]),"")  i\_subplot=i\_subplot+1  if it>198:  plt.subplot(2,3,i\_subplot)  show\_mixgauss\_prm3(X,Pi2,Mu2,Sigma2,Ratio4)  plt.title("{0:d}".format(it+1))  plt.xticks(range(X\_range0[0],X\_range0[1]),"")  plt.yticks(range(X\_range1[0],X\_range1[1]),"")  i\_subplot=i\_subplot+1  plt.show()  plt.figure(2,figsize=(4,4))  plt.plot(np.arange(max\_it)+1,Err2,color='k',linestyle='-',marker='o')  plt.grid(True)  plt.show()  print(Mu2) |

VII-3-4-1-xii. 再計算後の混合ガウスモデルの形

|  |
| --- |
| #混合ガウス関数（等高線と3d)  #等高線図と3d  Fig=plt.figure(1,figsize=(8,3.5))  Fig.add\_subplot(1,2,1)  show\_contour\_mixgauss(Pi2,Mu2,Sigma2)  plt.grid(True)  Ax=Fig.add\_subplot(1,2,2,projection='3d')  show3d\_mixgauss(Ax,Pi2, Mu2,Sigma2)  Ax.set\_xlabel('$x\_0$',fontsize=14)  Ax.set\_ylabel('$x\_1$',fontsize=14)  Ax.view\_init(40,-60)  plt.xlim(X\_range0)  plt.ylim(X\_range1)  plt.show() |

VII-3-4-1-xiii. 入力データが各クラスに属する確率

|  |
| --- |
| #個々入力データの尤度  xx=X  Gamma2,Ppk2,Ppl2,Ratio2=likelihood(xx,Mu2,Pi2,Sigma2)  print(Ppl2)  print(Ratio2) |

VII-3-4-1-xiv.再計算結果の図示（VII-3-4-xと同じ）

|  |
| --- |
| #結果を色分け図で表示する。  #表示する軸を選択する  #変数の選択  dim1=1  dim2=2  x\_range=[-2,2] #項目1の範囲  y\_range=[-2,2] #項目2の範囲  d0=dim1-1  d1=dim2-1  plt.figure(2,figsize=(4,4))  show\_mixgauss\_prm4(X,Pi2,Mu2,Ratio2)  plt.show() |

VII-3-4-1-xv. 任意の座標の尤度と所属確率

|  |
| --- |
| #データの尤度の評価  #評価するデータの入力  xxx=np.array([0.2,0.25])  g=np.zeros((K))  ppk3=np.zeros((K))  ppl3=np.zeros((K))  c1=-(D/2)\*np.log(2\*np.pi)  for k in range(K):  det\_sigma=np.linalg.det(Sigma2[k])  c2=-(1/2)\*np.log(det\_sigma)  inv\_sigma=np.linalg.inv(Sigma2[k])  c3=xxx-Mu2[k]  c4=np.dot(c3,inv\_sigma)  c5=np.dot(c4,c3)  c5=-c5/2  p=c1+c2+c5  g[k]=np.exp(p)  ppk3[k]=Pi2[k]\*g[k]  S=0  for k in range(K):  S=S+ppk3[k]  for k in range(K):  ppl3[k]=g[k]\*ppk3[k]/S  print(ppl3)  ratio3=np.zeros((K))  SS=0  for k in range(K):  SS=SS+ppl3[k]  for k in range(K):  ratio3[k]=ppl3[k]/SS  print(ratio3) |

VII-3-4-3-xvi. 結果の保存

|  |
| --- |
| #結果の保存  df=pd.DataFrame(Err2)  df.to\_csv('Error2\_r.csv')  df=pd.DataFrame(Pi2)  df.to\_csv('Pi2.csv')  df=pd.DataFrame(Mu2)  df.to\_csv('Centers2.csv')  df=pd.DataFrame(Gamma2)  df.to\_csv('Gamma2.csv')  df=pd.DataFrame(Ppl2)  df.to\_csv('Likelihood2.csv')  df=pd.DataFrame(Ratio2)  df.to\_csv('predict-P2.csv')  print(Sigma2) |