*VII-2-2.1層のニューラルネットワークモデル（ソフトマックス関数）*

VII-2-2-i. 準備とデータの読み込み

|  |
| --- |
| #リスト2-1.準備・データ読み込み  #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  #データの読み込み  df =pd.read\_csv("allometry.csv")  xn,D=df.shape  K=3 #クラスの数入力  D=D-K #説明変数の数  D1=D+1 #定数項を加えた変数ベクトルのディメンジョン  X=np.zeros((xn,D))  T3=np.zeros((xn,K))  for i in range(D):  X[:,i]=df.iloc[:,3+i]  for i in range(K):  T3[:,i]=df.iloc[:,i] |

VII-2-2-iiデータ分布の確認

|  |
| --- |
| *#色の指定*  X\_col=["b","r","g"]  #Scatterの内容  def show\_data(x,t):  ni,nj=t.shape #行数・列数を取得  for j in range (nj): plt.plot(x[t[:,j]==1,0],x[t[:,j]==1,1],linestyle="none",markeredgecolor="black",marker="o",color=X\_col[j],)  plt.grid(True)  #データを代入して結果表示  fig1=plt.figure(figsize=(4,4))  show\_data(X,T3)  plt.show() |

VII-2-2-iii. 関数の定義と実行

|  |
| --- |
| #softmax関数の定義  def softmax(x0,x1,w):  w=w.reshape((K,D1)) #wを３行３列の配列に並び替える  n=len(x1)#データの個数  y=np.zeros((n,K)) #配列の枠をつくる  for k in range(K):  y[:,k]=np.exp(w[k,0]+w[k,1]\*x0+w[k,2]\*x1)  wk=np.sum(y,axis=1) #行方向に総和を求める  wk=y.T/wk #の転置行列にwkの逆数の行列をかける  y=wk.T #転置行列を元に戻す  return y  #交差エントロピー誤差の定義  def cee\_softmax(w,x,t):  X\_n=x.shape[0] #データの個数  y=softmax(x[:,0],x[:,1],w)  cee=0  N,K=y.shape  for n in range(N):  for k in range (K):  cee=cee-(t[n,k]\*np.log(y[n,k]))  cee=cee/X\_n  return cee  #交差エントロピー誤差の偏微分の定義  def dcee\_softmax(w,x,t):  X\_n=x.shape[0]  y=softmax(x[:,0],x[:,1],w)  N,K=y.shape  dcee=np.zeros((K,D1))#クラスの数I,xの次元＋１（定数項）  for n in range(N):  for k in range(K):  dcee[k,:]=dcee[k,:]-(t[n,k]-y[n,k])\*np.r\_[1,x[n,:]]  dcee=dcee/X\_n  return dcee.reshape(-1)  #勾配降下法によるパラメータの推定  from scipy.optimize import minimize  def fit\_softmax(w\_init,x,t):  res=minimize(cee\_softmax,w\_init,args=(x,t),jac=dcee\_softmax,method='CG')  return res.x |

VII-2-2-iv.結果の図示

|  |
| --- |
| #等高線図の作成  def show\_contour\_softmax(w):  W\_init=np.zeros((K,D1))  W=fit\_softmax(W\_init,X,T3)  W=W.reshape((K,D1)) #WをK行D1列に配列  xn=211 #グリッドを作る  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)  z=np.zeros((K,xn,xn))  Z=np.zeros((xn,xn))  xxx0=xx0[0,:]  xxx1=xx1[:,0]  for i in range(xn):  for j in range(xn):  xy=([1,xxx0[j],xxx1[i]])  xy=np.reshape(xy,(1,3))  yx=xy.T  y=W.dot(yx)  e=np.exp(y)  u=np.sum(e,axis=0)  p=e/u  p=p.T  z[:,i,j]=p[:]  for k in range(K):  for i in range (xn):  for j in range (xn):  Z[i,j]=z[k,i,j]  cont=plt.contour(xxx0,xxx1,Z[:,:],levels=(0.5,0.95),colors=['cornflowerblue','k'])  cont.clabel(fmt='%.2f',fontsize=10)  plt.grid  #作図の実行  X\_range0=[140,210] #x0軸の範囲  X\_range1=[30,240] #x軸の範囲  x0g=30 #x0軸のグリッド数  x1g=30 #x1軸の議リッド数  x0=np.linspace(X\_range0[0],X\_range0[1],x0g)  x1=np.linspace(X\_range1[0],X\_range1[1],x1g)  W\_init=np.zeros((K,D1))  W=fit\_softmax(W\_init,X,T3)  print(np.round(W.reshape((K,D1)),3)) #3行３列で有効数字３けた  cee=cee\_softmax(W,X,T3)  print("cee={0:.3f}".format(cee))  fig2=plt.figure(figsize=(3,3))  show\_data(X,T3)  show\_contour\_softmax(W)  plt.show() |

VII-2-2-v.確率分布の3d表示

|  |
| --- |
| #確率分布の3d表現  from mpl\_toolkits.mplot3d import axes3d  #3d図の作成作業の定義  #3次元で確率をｚ軸にとる  def show3d\_softmax\_color(ax,w):  W\_init=np.zeros((3,3))  W=fit\_softmax(W\_init,X,T3)  W=W.reshape((3,3)) #Wを３行３列に配列  xn=211 #グリッドを作る  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)  z=np.zeros((K,xn,xn))  Z=np.zeros((xn,xn))  xxx0=xx0[0,:]  xxx1=xx1[:,0]  for i in range(xn):  for j in range(xn):  xy=([1,xxx0[j],xxx1[i]])  xy=np.reshape(xy,(1,3))  yx=xy.T  y=W.dot(yx)  e=np.exp(y)  u=np.sum(e,axis=0)  p=e/u  p=p.T  z[:,i,j]=p[:]  for i in range (xn):  for j in range (xn):  Z[i,j]=z[cluster,i,j]  y=Z  ax.plot\_surface(xx0,xx1,y,edgecolor="gray",rstride=20,cstride=20,alpha=0.3,shade=True,cmap="plasma")  #クラスターを選択する。  cluster=2  cluster=cluster-1  W=W.reshape((3,3))  Ax=plt.subplot(1,1,1,projection='3d')  fig3=plt.figure()  show3d\_softmax\_color(Ax,W) |

VII-2-2-vi. 3Dの散布図

|  |
| --- |
| #3dの散布図  from mpl\_toolkits.mplot3d import axes3d  #3d描画作業を定義する(2):上下に分けてデーター散布図を描く  def show\_data\_3d(ax,x,t):  for i in range (K):  if i==cluster:  level=1  if i!=cluster:  level=0  ax.plot(x[t[:,i]==1,0],x[t[:,i]==1,1],level,marker='o',color=X\_col[i],markeredgecolor='black',linestyle='none',markersize=5,alpha=0.5)  ax.view\_init(elev=25,azim=30)  #実行  fig4=plt.figure()  Ax=plt.subplot(projection='3d')  show\_data\_3d(Ax,X,T3)  show3d\_softmax\_color(Ax,W)  plt.show() |

VII-2-2-vii. 結果の保存

|  |
| --- |
| #推定されたパラメータ―  F=pd.DataFrame(W)  import pandas as pd  import csv  F.to\_csv('parameterBLBW1.csv')  #等高線入りの散布図  fig1.savefig('BLBWscatter.png')  #3dの確率とデータ  fig2.savefig('BLBWcontour.png')  #3確率  fig3.savefig('BLBW3d1.png') |