~\Documents\documents_general\structured_courses\math564\evaluations\projects\p04\solve4.m

```
 1
 2 % liver data analysis
 3
 4 % load data
 5 A=textread('liver.csv','%s','delimiter',',');          %#ok
 6 A=reshape(A,11,[]);
 7 [r,c]=size(A);
 8
 9 % ceate numerical data arrays
10 %   data is 10 by ~580 maximum normalized data values
11 %   class is 1 by ~580 0-1 classifier
12 data=zeros(r-1,c);
13 class=zeros(1,c);
14 idx=[];
15 for k=1:r
16     if k==2
17         for j=1:c
18             if ~isempty(A{2,j})
19                 data(k,j)=(length(A{2,j})-4)/2;
20             else
21                 idx=[idx j];                          %#ok
22             end
23         end
24     elseif k==11
25         for j=1:c
26             try
27                 class(j)=str2num(A{11,j})-1;          %#ok
28             catch
29                 idx=[idx j];                          %#ok
30             end
31         end
32     else
33         for j=1:c
34             try
35                 data(k,j)=str2num(A{k,j});            %#ok
36             catch
37                 idx=[idx j];                          %#ok
38             end
39         end
40     end
41 end
42
43 % remove instances with missing data
44 idx=unique(idx);
45 data(:,idx)=[];
46 class(idx)=[];
47
48 % normalize data to [0,1]
49 MinValues=min(data,[],2);
50 MaxValues=max(data,[],2);
51 data=(data-MinValues)./(MaxValues-MinValues);
```

```
52  [r,c]=size(data);
53
54  % subsample features
55  features=(1:10)';
56  data=data(features,:);
57  [r,c]=size(data);
58
59  % construct training data
60  m=100;
61  cdx=find(class==1);
62  ddx=find(class==0);
63  par.traindata=[data(:,cdx(1:m)) data(:,ddx(1:m))];
64  sh=1/3;
65  par.classdata=[class(cdx(1:m)) class(ddx(1:m))]*(1-2*sh)+sh;
66
67  % set up NN parameters and initial weights
68  par.dimensions=[10 10 10 1];
69  q=length(par.dimensions);
70  matrixsizes=par.dimensions(1:q-1).*par.dimensions(2:q);
71  numweights=sum(matrixsizes);
72  sz=1/sqrt(par.dimensions(1));
73  w=2*sz*rand(numweights,1)-sz;
74
75  % call the optimization
76  pr.objective=@nnloss;
77  pr.par=par;
78  pr.x0=w;
79  pr.method='BFGS';
80  pr.linesearch='StrongWolfe';
81  pr.maxiter=100000;
82  pr.dftol=1E-9;
83  pr.ngtol=1E-9;
84  pr.dxtol=1E-9;
85  pr.maxcond=1000;
86  pr.progress=1000;
87
88  % This is the call to the optimizer
89  pr.par.classify=false;
90  out=optimize(pr);
91
92  % This is a call to classify the training data
93  pr.par.classify=true;
94  res=pr.objective(out.x(:,end),pr.par);
95
96  % Show the confusion matrix for the training data
97  NN=sum(res<0.5&par.classdata<0.5);
98  NP=sum(res<0.5&par.classdata>=0.5);
99  PN=sum(res>=0.5&par.classdata<0.5);
100 PP=sum(res>=0.5&par.classdata>=0.5);
101 ConfusionMatrix=[NN PN ; NP PP]                %#ok
102
103 % Classify the test data and show the confusion matrix
104 pr.par.traindata=[data(:,cdx(m+1:end)) data(:,ddx(m+1:end))];
105 pr.par.classdata=[class(cdx(m+1:end)) class(ddx(m+1:end))]*1/3+1/3;
106 test=pr.objective(out.x(:,end),pr.par);
```

```
NN=sum(test<0.5&pr.par.classdata<0.5);
NP=sum(test<0.5&pr.par.classdata>=0.5);
PN=sum(test>=0.5&pr.par.classdata<0.5);
PP=sum(test>=0.5&pr.par.classdata>=0.5);
ConfusionMatrix=[NN PN ; NP PP]                %#ok


```