We need to develop a neural network based classifier for three various but related products. The collected data are the results of a chemical analysis of liquid products grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of products. The data file provided is in text format and has fourteen dimensions, the first of which determines the class of products (product ‘1’, product ‘2’, product ‘3’), which should serve as the output of the neural network classifier. The remaining ones determine the input of the classifier and has 13 constituents as:

1. Ethanol
2. Malic acid
3. Ash
4. Alcalinity of ash
5. Magnesium
6. Total phenols
7. Flavanoids
8. Nonflavanoid phenols
9. Proanthocyanins
10. Color intensity
11. Hue
12. OD280/OD315 of diluted liquid
13. Proline

1 Design a classifier (multilayer neural network), vary its parameters (number of hidden layers without exceeding three and number of nodes in each layer) and try to find the best possible classification performance (a table illustrating various results as parameters are varied would be preferred). Please discuss.

2 Once this is done, classify (determine to which product they belong) the following entries each of which has 13 attributes:

- **a)** 13.72; 1.43; 2.5; 16.7; 108; 3.4; 3.67; 0.19; 2.04; 6.8; 0.89; 2.87; 1285
- **b)** 12.04; 4.3; 2.38; 22; 80; 2.1; 1.75; 0.42; 1.35; 2.6; 0.79; 2.57; 580
- **c)** 14.13; 4.1; 2.74; 24.5; 96; 2.05; 0.76; 0.56; 1.35; 9.2; 0.61; 1.6; 560

Hint for implementation: You may wish to calibrate all input data to be all between 0 and one. Also from the set of data choose 75% of the data from product 1, product 2 and product 3 as training data and remaining 25% remaining as testing. You can use existing Matlab libraries or other libraries to create the classifier, which code needs to be appended to the solutions.

In this example, we will follow the same process we did in the CNN tutorial. We will implement a shallow MLP on the attached problem. In addition to the libraries we used before, we will also use pandas (http://pandas.pydata.org/ (http://pandas.pydata.org/)) to read the CSV file containing the data.

In [1]:

```
from __future__ import division
from __future__ import print_function

import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
```

```
Using TensorFlow backend.
```

In [2]:

```
data = pd.read_csv('data.txt', header=None)
print(data.head())
```

```
   0      1     2     3     4    5     6     7     8     9    10    
11    12  \
0   1  14.23  1.71  2.43  15.6  127  2.80  3.06  0.28  2.29  5.64  1
.04  3.92
1   1  13.20  1.78  2.14  11.2  100  2.65  2.76  0.26  1.28  4.38  1
.05  3.40
2   1  13.16  2.36  2.67  18.6  101  2.80  3.24  0.30  2.81  5.68  1
.03  3.17
3   1  14.37  1.95  2.50  16.8  113  3.85  3.49  0.24  2.18  7.80  0
.86  3.45
4   1  13.24  2.59  2.87  21.0  118  2.80  2.69  0.39  1.82  4.32  1
.04  2.93

     13
0  1065
1  1050
2  1185
3  1480
4   735
```

It is good practice to shuffle the data before you use it.

In [3]:

```
data = data.as_matrix()
np.random.shuffle(data)
```

Each row in the CSV file contains a data sample. In each row, the first entry represents the class (we have 3 classes), and the rest of the 13 entries are the features. So, we are going to split the data into labels and features. We are also going to standardize the data. In general, you should NOT standardize the data this way, but instead use only the mean and std estimated from the training set to standardize all datasets (this way you make sure results on the validation set are not optimistic compared to results you would get on future test sets).

In [4]:

```
y = data[:, 0:1]
X = data[:, 1:]
X = (X - np.mean(X, axis=0)) / np.std(X, axis=0)
```

The labels are given as integers '1', '2', and '3'. As we did in the CNN tutorial, we are going to convert them into one-hot encoding.

In [5]:

```
print(y[:10])
```

```
[[ 2.]
 [ 2.]
 [ 3.]
 [ 2.]
 [ 1.]
 [ 2.]
 [ 2.]
 [ 3.]
 [ 2.]
 [ 1.]]
```

In [6]:

```
y = to_categorical(y-1)
print(y[:10])
```

```
[[ 0.  1.  0.]
 [ 0.  1.  0.]
 [ 0.  0.  1.]
 [ 0.  1.  0.]
 [ 1.  0.  0.]
 [ 0.  1.  0.]
 [ 0.  1.  0.]
 [ 0.  0.  1.]
 [ 0.  1.  0.]
 [ 1.  0.  0.]]
```

Finally, we are going to build our single hidden layer MLP and compile it, as we did with the CNN. When training the model, we are going to use 25% of the data for validation.

In [7]:

```
mlp = Sequential()
mlp.add(Dense(512, activation='sigmoid', input_dim=13))
mlp.add(Dense(3, activation='softmax'))
mlp.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy
'])
```

In [9]:

```
mlp.fit(X, y, batch_size=32, epochs=100, validation_split=0.25, verbose=1)
```

```
Train on 132 samples, validate on 44 samples
Epoch 1/100
132/132 [==============================] - 0s - loss: 0.8931 - acc:
0.8636 - val_loss: 0.9146 - val_acc: 0.4091
Epoch 2/100
132/132 [==============================] - 0s - loss: 0.8894 - acc:
0.6742 - val_loss: 0.8713 - val_acc: 0.4773
Epoch 3/100
132/132 [==============================] - 0s - loss: 0.8586 - acc:
0.6515 - val_loss: 0.8474 - val_acc: 0.6591
Epoch 4/100
132/132 [==============================] - 0s - loss: 0.8590 - acc:
0.8636 - val_loss: 0.8774 - val_acc: 0.5227
Epoch 5/100
132/132 [==============================] - 0s - loss: 0.8401 - acc:
0.7803 - val_loss: 0.8288 - val_acc: 0.7045
Epoch 6/100
132/132 [==============================] - 0s - loss: 0.8125 - acc:
0.7955 - val_loss: 0.8214 - val_acc: 0.9545
Epoch 7/100
132/132 [==============================] - 0s - loss: 0.8031 - acc:
0.9167 - val_loss: 0.8680 - val_acc: 0.5000
Epoch 8/100
132/132 [==============================] - 0s - loss: 0.8080 - acc:
0.8561 - val_loss: 0.8661 - val_acc: 0.4091
Epoch 9/100
132/132 [==============================] - 0s - loss: 0.7931 - acc:
0.6061 - val_loss: 0.7819 - val_acc: 0.9545
Epoch 10/100
132/132 [==============================] - 0s - loss: 0.7578 - acc:
0.9318 - val_loss: 0.7612 - val_acc: 0.7500
Epoch 11/100
132/132 [==============================] - 0s - loss: 0.7522 - acc:
0.8788 - val_loss: 0.7528 - val_acc: 0.7045
Epoch 12/100
132/132 [==============================] - 0s - loss: 0.7401 - acc:
0.7879 - val_loss: 0.7781 - val_acc: 0.8636
Epoch 13/100
132/132 [==============================] - 0s - loss: 0.7322 - acc:
0.8864 - val_loss: 0.7399 - val_acc: 0.9545
Epoch 14/100
```

```
132/132 [==============================] - 0s - loss: 0.7083 - acc:
0.9697 - val_loss: 0.7378 - val_acc: 0.7955
Epoch 15/100
132/132 [==============================] - 0s - loss: 0.7081 - acc:
0.8939 - val_loss: 0.7089 - val_acc: 0.9545
Epoch 16/100
132/132 [==============================] - 0s - loss: 0.6848 - acc:
0.9697 - val_loss: 0.7005 - val_acc: 0.9091
Epoch 17/100
132/132 [==============================] - 0s - loss: 0.6757 - acc:
0.9621 - val_loss: 0.6996 - val_acc: 0.7955
Epoch 18/100
132/132 [==============================] - 0s - loss: 0.6698 - acc:
0.9394 - val_loss: 0.7018 - val_acc: 0.8182
Epoch 19/100
132/132 [==============================] - 0s - loss: 0.6576 - acc:
0.9091 - val_loss: 0.6802 - val_acc: 0.9545
Epoch 20/100
132/132 [==============================] - 0s - loss: 0.6463 - acc:
0.9621 - val_loss: 0.6605 - val_acc: 0.8864
Epoch 21/100
132/132 [==============================] - 0s - loss: 0.6322 - acc:
0.9394 - val_loss: 0.6564 - val_acc: 0.9318
Epoch 22/100
132/132 [==============================] - 0s - loss: 0.6266 - acc:
0.9470 - val_loss: 0.6463 - val_acc: 0.9318
Epoch 23/100
132/132 [==============================] - 0s - loss: 0.6124 - acc:
0.9697 - val_loss: 0.6332 - val_acc: 0.9545
Epoch 24/100
132/132 [==============================] - 0s - loss: 0.6065 - acc:
0.9697 - val_loss: 0.6249 - val_acc: 0.9318
Epoch 25/100
132/132 [==============================] - 0s - loss: 0.5919 - acc:
0.9470 - val_loss: 0.6194 - val_acc: 0.9545
Epoch 26/100
132/132 [==============================] - 0s - loss: 0.5897 - acc:
0.9697 - val_loss: 0.6206 - val_acc: 0.9091
Epoch 27/100
132/132 [==============================] - 0s - loss: 0.5783 - acc:
0.9545 - val_loss: 0.6027 - val_acc: 0.9545
Epoch 28/100
132/132 [==============================] - 0s - loss: 0.5701 - acc:
0.9773 - val_loss: 0.5967 - val_acc: 0.8864
Epoch 29/100
132/132 [==============================] - 0s - loss: 0.5571 - acc:
0.9545 - val_loss: 0.6307 - val_acc: 0.8409
Epoch 30/100
132/132 [==============================] - 0s - loss: 0.5683 - acc:
0.9242 - val_loss: 0.5843 - val_acc: 0.9318
Epoch 31/100
132/132 [==============================] - 0s - loss: 0.5518 - acc:
0.9394 - val_loss: 0.5759 - val_acc: 0.9545
```

```
Epoch 32/100
132/132 [==============================] - 0s - loss: 0.5374 - acc:
0.9773 - val_loss: 0.5682 - val_acc: 0.9773
Epoch 33/100
132/132 [==============================] - 0s - loss: 0.5326 - acc:
0.9697 - val_loss: 0.5698 - val_acc: 0.9545
Epoch 34/100
132/132 [==============================] - 0s - loss: 0.5192 - acc:
0.9848 - val_loss: 0.5509 - val_acc: 0.9773
Epoch 35/100
132/132 [==============================] - 0s - loss: 0.5130 - acc:
0.9773 - val_loss: 0.5466 - val_acc: 0.9773
Epoch 36/100
132/132 [==============================] - 0s - loss: 0.5046 - acc:
0.9848 - val_loss: 0.5579 - val_acc: 0.9318
Epoch 37/100
132/132 [==============================] - 0s - loss: 0.5033 - acc:
0.9773 - val_loss: 0.5329 - val_acc: 0.9773
Epoch 38/100
132/132 [==============================] - 0s - loss: 0.4896 - acc:
0.9773 - val_loss: 0.5338 - val_acc: 0.9318
Epoch 39/100
132/132 [==============================] - 0s - loss: 0.4917 - acc:
0.9773 - val_loss: 0.5225 - val_acc: 0.9545
Epoch 40/100
132/132 [==============================] - ETA: 0s - loss: 0.4808 -
acc: 1.000 - 0s - loss: 0.4848 - acc: 0.9697 - val_loss: 0.5151 - va
l_acc: 0.9773
Epoch 41/100
132/132 [==============================] - 0s - loss: 0.4720 - acc:
0.9773 - val_loss: 0.5087 - val_acc: 0.9773
Epoch 42/100
132/132 [==============================] - 0s - loss: 0.4692 - acc:
0.9773 - val_loss: 0.5033 - val_acc: 0.9545
Epoch 43/100
132/132 [==============================] - 0s - loss: 0.4586 - acc:
0.9545 - val_loss: 0.5070 - val_acc: 0.9318
Epoch 44/100
132/132 [==============================] - 0s - loss: 0.4623 - acc:
0.9697 - val_loss: 0.5080 - val_acc: 0.9091
Epoch 45/100
132/132 [==============================] - 0s - loss: 0.4496 - acc:
0.9697 - val_loss: 0.4873 - val_acc: 0.9773
Epoch 46/100
132/132 [==============================] - 0s - loss: 0.4403 - acc:
0.9773 - val_loss: 0.4891 - val_acc: 0.9545
Epoch 47/100
132/132 [==============================] - 0s - loss: 0.4350 - acc:
0.9848 - val_loss: 0.4777 - val_acc: 0.9773
Epoch 48/100
132/132 [==============================] - 0s - loss: 0.4302 - acc:
0.9773 - val_loss: 0.4782 - val_acc: 0.9545
Epoch 49/100
```

```
132/132 [==============================] - 0s - loss: 0.4252 - acc:
0.9773 - val_loss: 0.4753 - val_acc: 0.9318
Epoch 50/100
132/132 [==============================] - 0s - loss: 0.4227 - acc:
0.9848 - val_loss: 0.4853 - val_acc: 0.8864
Epoch 51/100
132/132 [==============================] - 0s - loss: 0.4183 - acc:
0.9621 - val_loss: 0.4567 - val_acc: 0.9773
Epoch 52/100
132/132 [==============================] - 0s - loss: 0.4124 - acc:
0.9697 - val_loss: 0.4545 - val_acc: 0.9545
Epoch 53/100
132/132 [==============================] - 0s - loss: 0.4064 - acc:
0.9545 - val_loss: 0.4483 - val_acc: 0.9545
Epoch 54/100
132/132 [==============================] - 0s - loss: 0.4018 - acc:
0.9773 - val_loss: 0.4482 - val_acc: 0.9545
Epoch 55/100
132/132 [==============================] - 0s - loss: 0.3951 - acc:
0.9773 - val_loss: 0.4469 - val_acc: 0.9091
Epoch 56/100
132/132 [==============================] - 0s - loss: 0.3935 - acc:
0.9545 - val_loss: 0.4349 - val_acc: 0.9773
Epoch 57/100
132/132 [==============================] - 0s - loss: 0.3860 - acc:
0.9848 - val_loss: 0.4309 - val_acc: 0.9773
Epoch 58/100
132/132 [==============================] - 0s - loss: 0.3806 - acc:
0.9848 - val_loss: 0.4301 - val_acc: 0.9545
Epoch 59/100
132/132 [==============================] - 0s - loss: 0.3766 - acc:
0.9773 - val_loss: 0.4283 - val_acc: 0.9545
Epoch 60/100
132/132 [==============================] - 0s - loss: 0.3777 - acc:
0.9697 - val_loss: 0.4255 - val_acc: 0.9545
Epoch 61/100
132/132 [==============================] - 0s - loss: 0.3727 - acc:
0.9848 - val_loss: 0.4153 - val_acc: 0.9773
Epoch 62/100
132/132 [==============================] - 0s - loss: 0.3651 - acc:
0.9697 - val_loss: 0.4149 - val_acc: 0.9545
Epoch 63/100
132/132 [==============================] - 0s - loss: 0.3635 - acc:
0.9697 - val_loss: 0.4220 - val_acc: 0.8864
Epoch 64/100
132/132 [==============================] - 0s - loss: 0.3634 - acc:
0.9545 - val_loss: 0.4058 - val_acc: 0.9773
Epoch 65/100
132/132 [==============================] - 0s - loss: 0.3522 - acc:
0.9848 - val_loss: 0.4051 - val_acc: 0.9545
Epoch 66/100
132/132 [==============================] - 0s - loss: 0.3539 - acc:
0.9848 - val_loss: 0.4019 - val_acc: 0.9773
```

```
Epoch 67/100
132/132 [==============================] - 0s - loss: 0.3471 - acc:
0.9848 - val_loss: 0.4052 - val_acc: 0.9545
Epoch 68/100
132/132 [==============================] - 0s - loss: 0.3445 - acc:
0.9848 - val_loss: 0.4050 - val_acc: 0.9545
Epoch 69/100
132/132 [==============================] - 0s - loss: 0.3431 - acc:
0.9773 - val_loss: 0.3885 - val_acc: 0.9773
Epoch 70/100
132/132 [==============================] - 0s - loss: 0.3343 - acc:
0.9773 - val_loss: 0.3854 - val_acc: 0.9545
Epoch 71/100
132/132 [==============================] - 0s - loss: 0.3314 - acc:
0.9848 - val_loss: 0.3856 - val_acc: 0.9773
Epoch 72/100
132/132 [==============================] - 0s - loss: 0.3349 - acc:
0.9773 - val_loss: 0.3976 - val_acc: 0.9545
Epoch 73/100
132/132 [==============================] - 0s - loss: 0.3325 - acc:
0.9773 - val_loss: 0.3793 - val_acc: 0.9773
Epoch 74/100
132/132 [==============================] - 0s - loss: 0.3210 - acc:
0.9848 - val_loss: 0.3842 - val_acc: 0.9318
Epoch 75/100
132/132 [==============================] - 0s - loss: 0.3242 - acc:
0.9697 - val_loss: 0.3764 - val_acc: 0.9545
Epoch 76/100
132/132 [==============================] - 0s - loss: 0.3185 - acc:
0.9773 - val_loss: 0.3723 - val_acc: 0.9773
Epoch 77/100
132/132 [==============================] - 0s - loss: 0.3121 - acc:
0.9848 - val_loss: 0.3665 - val_acc: 0.9773
Epoch 78/100
132/132 [==============================] - 0s - loss: 0.3110 - acc:
0.9848 - val_loss: 0.3617 - val_acc: 0.9773
Epoch 79/100
132/132 [==============================] - 0s - loss: 0.3091 - acc:
0.9924 - val_loss: 0.3591 - val_acc: 0.9773
Epoch 80/100
132/132 [==============================] - 0s - loss: 0.3050 - acc:
0.9773 - val_loss: 0.3604 - val_acc: 0.9545
Epoch 81/100
132/132 [==============================] - 0s - loss: 0.3019 - acc:
0.9848 - val_loss: 0.3547 - val_acc: 0.9545
Epoch 82/100
132/132 [==============================] - 0s - loss: 0.2978 - acc:
0.9848 - val_loss: 0.3498 - val_acc: 0.9773
Epoch 83/100
132/132 [==============================] - 0s - loss: 0.2978 - acc:
0.9773 - val_loss: 0.3479 - val_acc: 0.9773
Epoch 84/100
132/132 [==============================] - 0s - loss: 0.2939 - acc:
```

```
0.9697 - val_loss: 0.3541 - val_acc: 0.9545
Epoch 85/100
132/132 [==============================] - 0s - loss: 0.2967 - acc:
0.9773 - val_loss: 0.3434 - val_acc: 0.9773
Epoch 86/100
132/132 [==============================] - 0s - loss: 0.2877 - acc:
0.9773 - val_loss: 0.3421 - val_acc: 0.9773
Epoch 87/100
132/132 [==============================] - 0s - loss: 0.2857 - acc:
0.9848 - val_loss: 0.3394 - val_acc: 0.9545
Epoch 88/100
132/132 [==============================] - 0s - loss: 0.2839 - acc:
0.9848 - val_loss: 0.3492 - val_acc: 0.9318
Epoch 89/100
132/132 [==============================] - 0s - loss: 0.2849 - acc:
0.9773 - val_loss: 0.3323 - val_acc: 0.9773
Epoch 90/100
132/132 [==============================] - 0s - loss: 0.2785 - acc:
0.9773 - val_loss: 0.3324 - val_acc: 0.9773
Epoch 91/100
132/132 [==============================] - 0s - loss: 0.2757 - acc:
0.9773 - val_loss: 0.3414 - val_acc: 0.9545
Epoch 92/100
132/132 [==============================] - 0s - loss: 0.2780 - acc:
0.9848 - val_loss: 0.3355 - val_acc: 0.9773
Epoch 93/100
132/132 [==============================] - 0s - loss: 0.2719 - acc:
0.9848 - val_loss: 0.3463 - val_acc: 0.9545
Epoch 94/100
132/132 [==============================] - 0s - loss: 0.2770 - acc:
0.9773 - val_loss: 0.3301 - val_acc: 0.9545
Epoch 95/100
132/132 [==============================] - 0s - loss: 0.2689 - acc:
0.9773 - val_loss: 0.3222 - val_acc: 0.9773
Epoch 96/100
132/132 [==============================] - 0s - loss: 0.2649 - acc:
0.9848 - val_loss: 0.3228 - val_acc: 0.9773
Epoch 97/100
132/132 [==============================] - 0s - loss: 0.2646 - acc:
0.9773 - val_loss: 0.3194 - val_acc: 0.9773
Epoch 98/100
132/132 [==============================] - 0s - loss: 0.2607 - acc:
0.9848 - val_loss: 0.3300 - val_acc: 0.9318
Epoch 99/100
132/132 [==============================] - 0s - loss: 0.2654 - acc:
0.9697 - val_loss: 0.3151 - val_acc: 0.9773
Epoch 100/100
132/132 [==============================] - 0s - loss: 0.2541 - acc:
0.9848 - val_loss: 0.3116 - val_acc: 0.9773
```

Out[9]:

```
<keras.callbacks.History at 0x7fa4c3e03128>
```