# HOMEWORK 2 SOLUTIONS - NEURAL COMPUTATION

## Problem 1)

The implementation of the pocket algorithm can be found in the appendix. The double-moon dataset is here (I made some modification in the data generation process).

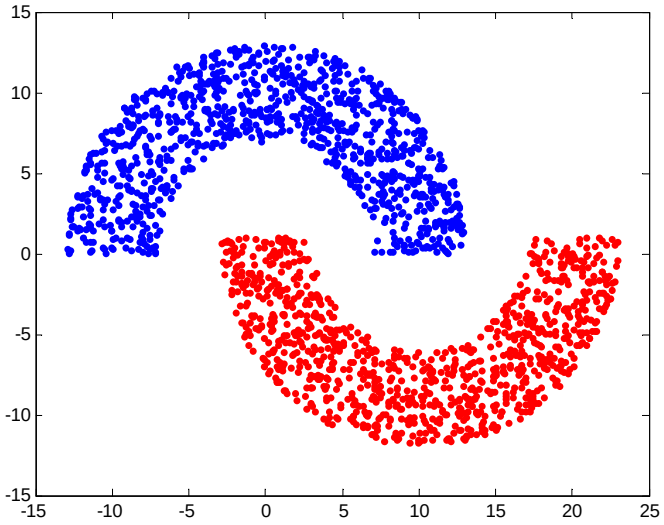


Fig 1: Double-moon dataset for d = -1

Here is the Decision boundary obtained by applying pocket algorithm on the above dataset with comparison to result obtained from perceptron algorithm:

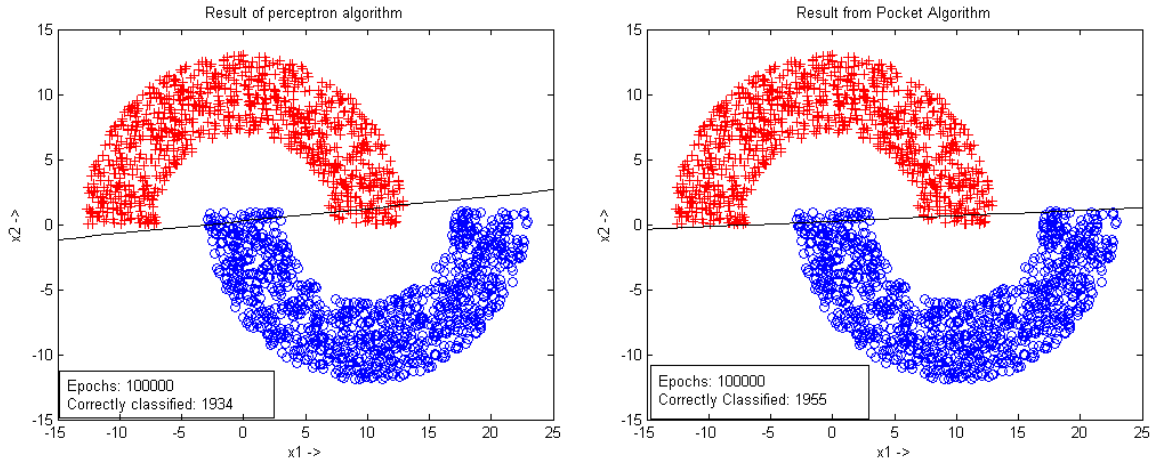


Fig 2: Performances of Pocket algorithm over Perceptron algorithm for d = -1. Perceptron classifies 1934 points correctly whereas pocket classifies 1955 points correctly.

**Problem 2)**

A feed forward Neural network was implemented using MATLAB NN toolbox. A two layer NN with 4 hidden layers were implemented. The training data was normalized and fed to the network to be fitted. NN was trained using back-propagation algorithm. Here is the classification result.

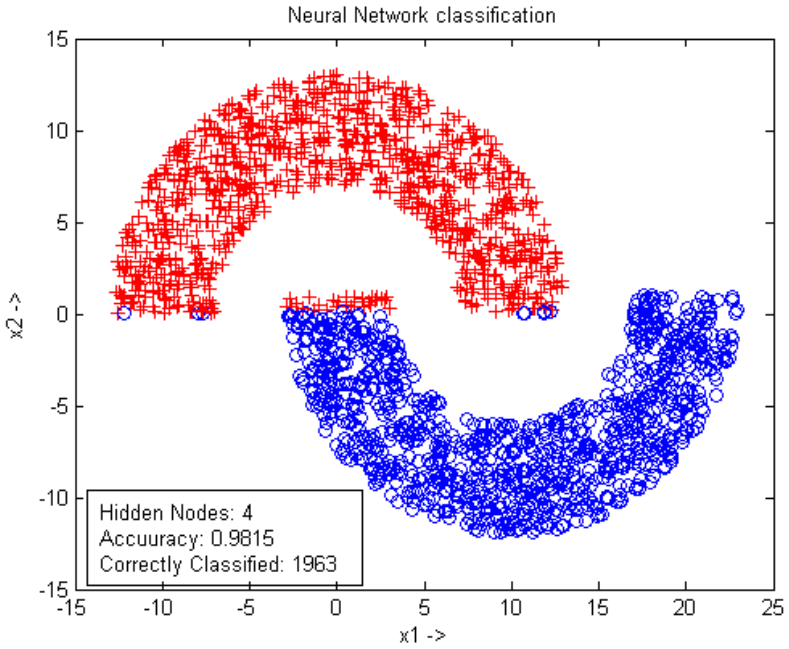


Fig 4: Classification with Neural Network. 4 hidden layers produces an accuracy of 98.15%. 6 hidden produces 100% accuracy.

**Problem 3)**

a) *Data Preprocessing:* No data cleaning was required. The first feature ID was ignored because it is an artificially generated number, having no relation with the actual data. Second feature is the target variable which was mapped in {1,-1} using following rule obtained from data description file:

I. If column2 == 'R' and column3 < 24 then class = 1
II. If column2 == 'N' and column3 >= 24 then class = 0
III Else ignore that record.

Using the above rules, the number of data points were reduced to 140 from 198.

b) *Dimensionality reduction:* There were 31 features in dataset and only 140 data points (Only 112 will be used as training set in each iteration). So it was required to apply some dimensionality reduction technique in order to build a good classifier.

We found there were a huge correlation among some of the features (See figure 5). So we removed the features having correlation more than 0.9 with other features. Then we applied PCA on the remaining features to reduce the number of attributes to 9 (Still 95% of the original variance was retained).

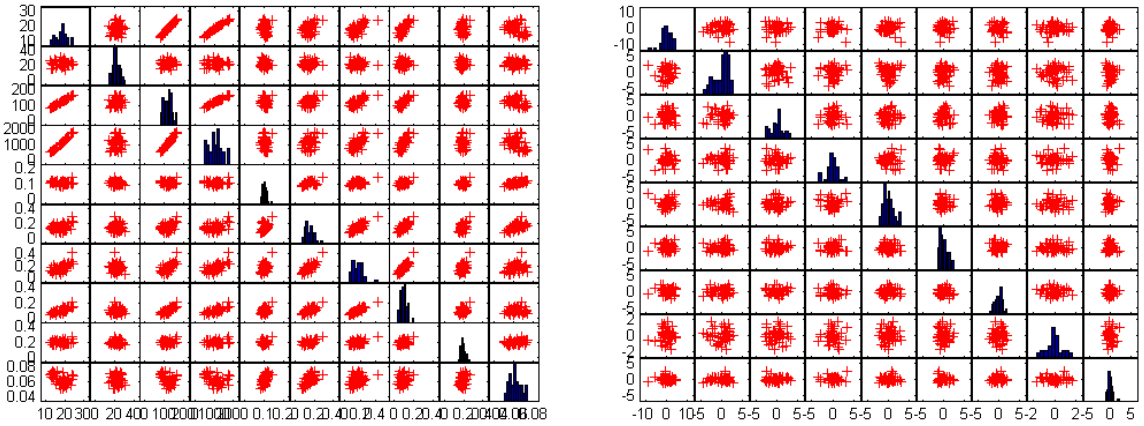

Fig 5: Correlation matrix of features before (10 out of 31) and after (9 out of 9) dimensionality reduction

c) *Training*: 5 Cross validation was performed by dividing the entire dataset in 5 equal parts. In each iteration 20% (one part) was used as test data and the NN was trained with the rest 80% (four parts) after normalization. The average accuracy for 5 such iterations were reported for different number of hidden nodes (5 to 10), learning rate (e-4 to e0 with a step of e-0.5) and momentum (1 to 0.5 with a step of -0.1). Following result was obtained:

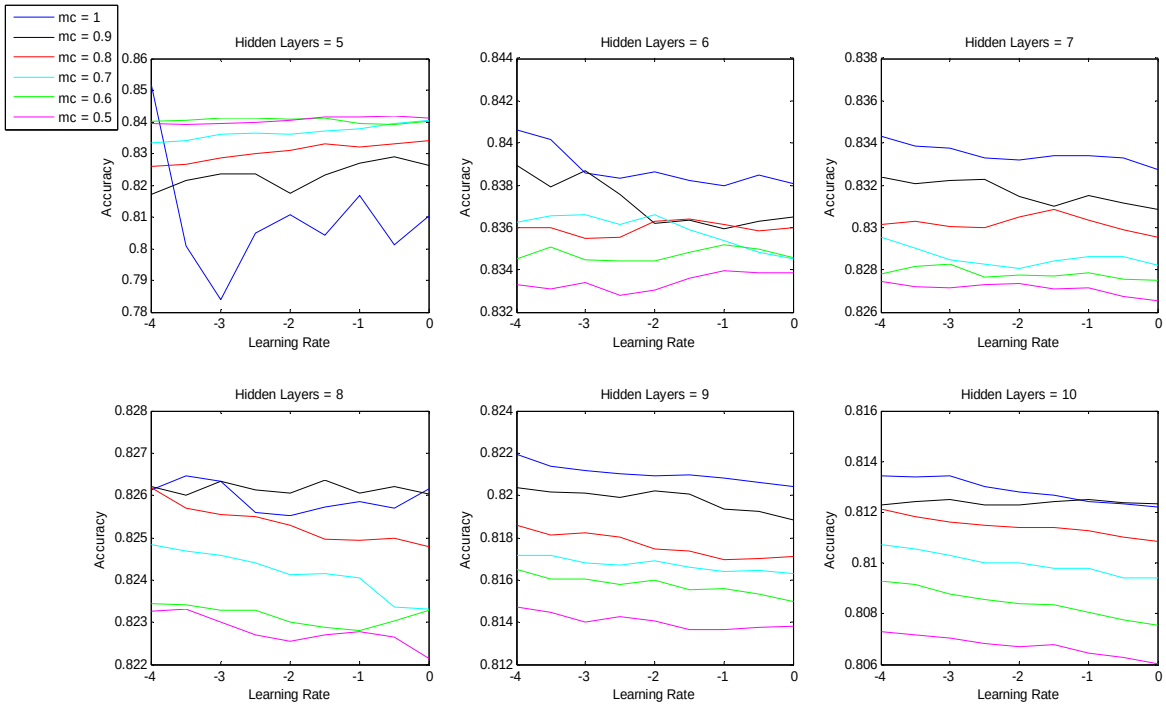


Fig 6: Accuracy Obtained with different number of hidden layers, learning rate and momentum parameters.

From the above figure it seems there is little variation in accuracy for different hyper-parameters. Best result is obtained for following combination:

*Best combination:*
No. hidden layers = 5
Learning Rate = 0.0001
Momentum = 1
Accuracy = 0.85

**Problem 4)**

There is only one difference between this problem and problem 3. The target variable in last problem was a binary class and that in this problem is a continuous variable (Classification vs. Regression problem). Everything else was same. We followed the same preprocessing and dimensionality reduction schemes in this problem and reported the $R^2$ accuracy instead of fraction correct. But we could only obtain an extremely poor accuracy (-0.485).

The reason for this poor performance might be the multimodal distribution of target variable which is given in fig 7.

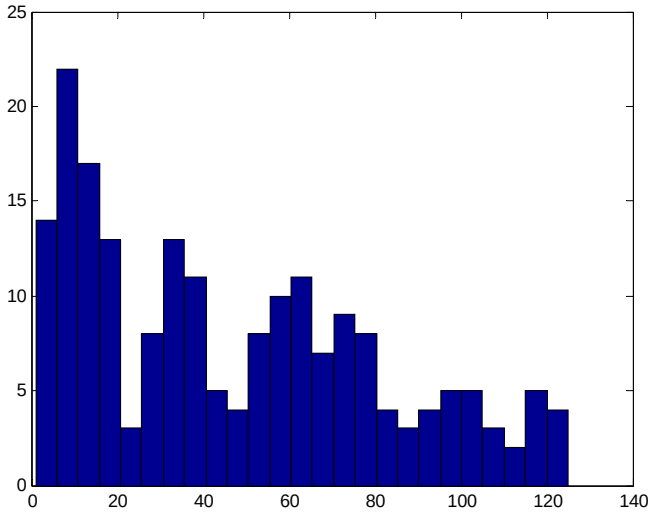


Fig 7: Histogram of time to recur

From the data description it was found that this target variable has different meaning depending on different values of column 2 (Recurrent/non-recurrent). For recurrent type this is recur time and for non-recurrent type this is disease-free time. So we split the data into two parts based on N/R values of column 1 and applied NN. We got following accuracies:

| *Recurrence* | *Accuracy* |
|---|---|
| R | -0.166 |
| N | -0.8326 |

So, the accuracy was still not satisfactory. So we conclude →

1) Either there is no correlation between the attributes and targets, i.e. the target is not learnable.

2) Or, The functional relation between attributes and the target is too complex and need more training data to accurately learn the function.

## APPENDIX

### Pocket Algorithm:

```
x = [ones(2000,1) dataset(:,1:2)];

lr = 0.1;                   % Learning Rate
w = [0 0 0]';               % Initial Weight
pocketW = w;                % Initial weights in pocket;
X1 = -15:25;
epoch = 1;
acc = sum(dataset(:,3)'==((w'*x'>= 0)+(w'*x'>= 0)-1));      %Accuracy of
the machine at hand
accPocket(epoch) = acc;
%Accuracy of the machine in pocket

while(accPocket(epoch)~=2000)
    i=mod(epoch,size(dataset,1))+1;
    f = (w'*x(i,1:3)'>=0)+((w'*x(i,1:3)'>=0)-1);
    w = w + lr*(dataset(i,3) - f)*x(i,1:3)';
    acc = sum(dataset(:,3)'==((w'*x'>= 0)+(w'*x'>= 0)-1));
    if(acc > accPocket(epoch))
        pocketW = w;
        accPocket(epoch+1) = acc;
    else
        accPocket(epoch+1) = accPocket(epoch);
    end

    epoch=epoch+1;
    if(epoch==100000)
        break;
    end
end

finalW = pocketW;
plot(X1,(-finalW(1)/finalW(3))-(finalW(2)/finalW(3))*X1,'k-');
disp(sprintf('Solution reached in %d epochs', epoch));
disp(sprintf('%d point classified correctly',accPocket(end)));
```