
 Practical/Assignment 2

2.1 Linear Networks — Adaline and its applications

1. Study the following demo files which demonstrate properties and applications of linear neural networks:

`$CSE5301/Mtlb/adln1.m`, `$NNET/nndemos/demolin1...8.m`.

2. Consider a problem of approximation of a collection of points by a **regression line**. In this case the dimensionality of the augmented input space is $p = 2$. The objective is to find the regression line: $y = a \cdot x + b$, that is, to find the parameters $\{a, b\}$ specifying the line, from the collection of points $\{\mathbf{x}(n), d(n)\}$.

- (a) Re-write the Normal (Wiener-Hopf) equation directly for two unknown weights, w_1, w_2 and points, $\{x_1(n), x_2(n) = 1, d(n)\}$ for $n = 1, \dots, N$. Derive equations for $a = w_1, b = w_2$ as functions of the training points. In the derived expressions identify the following statistical quantities: the mean, \bar{x} the second moment \bar{x}^2 of \mathbf{x} , the mean \bar{d} of d , and the correlation c of d and x .
- (b) Illustrate using MATLAB the problem of fitting the regression line into a collection of points from a $\{x, y\}$ plane. Compare the results obtained using the following three methods:
 - i. Direct solution of the Wiener-Hopf equation as derived above,
 - ii. The LMS learning law,
 - iii. The Sequential Regression algorithm.

3. Study the following MATLAB scripts which demonstrate applications of linear neural networks in adaptive signal processing:

`$CSE5301/Mtlb/adlpr.m`, `adsid.m`, `adlnc.m`

`$NNET/nndemos/applin1...4.m`

4. Modify the following scripts

- (a) `adlpr.m` — adaptive prediction,
- (b) `adsid.m` — adaptive system identification,
- (c) `adlnc.m` — adaptive noise cancellation

Modification should include:

- replacement of the LMS learning law by the sequential regression (SR) algorithm,
- usage of the amplitude and frequency modulated sinusoidal input signal in all three cases. Modify parameters of the signal and filters involved.

2.2 Multilayer Perceptron — Back-Propagation

1. Study the following demo files which demonstrate properties and applications of multilayer perceptrons and backpropagation learning law:

```
$CSE5301/Mt1b/fapr2D.m,  
$NNET/nnd12sd1, nnd12sd2, nnd12v1, nnd12cg, nnd12m
```

2. Write a function approximation script similar to `fapr2D.m` with the following modifications:

- (a) The function to be approximated, $d = f(x)$, is of the following form:

$$d = 0.02x(x^2 - 13x + 48) \cos(2x) + 0.2\text{noise}, \text{ for } x \in (0, \dots, 9)$$

Note that to plot such a function in MATLAB you can use:

```
x = 0:dx:9 ;  
d = 0.02*polyval([1 -13 48 0],x).*cos(2*x)+0.2*rand(1,N);  
plot(x, d), grid
```

- (b) Use the **pattern update** mode in learning
 - (c) Use the Nguyen-Widrow initialisation.
3. Repeat the above exercise using the Neural Network Toolbox. Use the **conjugate gradient** and **Levenberg-Marquardt** learning laws and compare their efficiency.
 4. For the above neural network plot **two** error surfaces, $J(w_a, w_b)$, where (w_a, w_b) is a pair of weights of your choice from the hidden and output layers.
 5. Use a two-layer perceptron and a selected backpropagation learning law (conjugate gradient or Levenberg-Marquardt) to design an XOR gate. Aim at the minimal total number of synapses.

Once the learning is finished, replace the output activation function with a unipolar step function and verify that the perceptron really acts as an XOR gate.

Sketch the detailed structure of the obtained XOR perceptron.

2.3 Image coding algorithm with a two-layer perceptron

Implement in MATLAB an image coding algorithm using a multi-layer perceptron and an advanced learning algorithm:

- Demo images can be found in

`$MATLAB/toolbox/matlab/demos/*.mat`

To examine the demo images run:

```
load file_name
whos
image(X),
colormap(map), title(caption)
imageext
```

- For initial experimentation use a 60×80 pixel sub-image from any image available in MATLAB.
- The block-matrix conversion functions `blkM2vc.m`, `vc2blkM.m` are in <http://www.csse.monash.edu.au/~app/CSE5301/Mtlb>
- Use either the Levenberg-Marquardt or a conjugate gradient algorithm.
- Determine the speed of training and the SNR ratio.

2.4 Image coding algorithm using the generalised Hebbian algorithm

Implement in MATLAB an image compression system using the generalised Hebbian learning algorithm (GHA) as described in lecture notes.

- Determine the speed of training and the SNR ratio. Compare it with the equivalent results for the multilayer perceptron.
- Demonstrate that the weight matrix W which is obtained by the GH learning algorithm is an approximation of the matrix of eigenvectors of the input correlation matrix, that is, $\hat{W} = \text{eig}(R)$.
- Demonstrate that the variance of the output signals is equal to the eigenvalues of the input correlation matrix.
- Compare briefly this image compression system to that based on the two-layer perceptron.

Your submission should include

- Brief comments regarding the demo files,
- Relevant derivations, equations, scripts and plots with suitable comments.