Dr. Qadri Hamarsheh

Neural Networks and Fuzzy Logic (630514)

Lecture 10

Supervised Learning in Neural Networks
(Part 3)

Supervised Learning in Neural Networks – using matlab

- The **MATLAB® Neural Network Toolbox** implements some of the most **popular training algorithms**, which encompass both original gradient-descent and faster training methods.
- **Batch Gradient Descent (traingd):**
    - Original but the slowest.
    - Weights and biases updated in the direction of the negative gradient.
    - Selected by setting **trainFcn** to **traingd**:

    net = newff(minmax(p), [3 1], {'tansig', 'purelin'}, 'traingd');
- **Batch Gradient Descent with Momentum (traingdm):**
    - Faster convergence than traingd.
    - Momentum allows the network to respond not only the local gradient, but also to recent trends in the error surface.
    - Momentum allows the network to ignore small features in the error surface; without momentum a network may get stuck in a shallow local minimum.
    - Selected by setting trainFcn to traingdm:

    net = newff (minmax(p), [3 1], {'tansig', 'purelin'}, 'traingdm');
    - Faster Training.
- The MATLAB® Neural Network Toolbox also implements some of the **faster training methods**, in which the training can converge from ten to one hundred times faster than **traingd** and **traingdm**.
    - These faster algorithms fall into two categories:
        1. **Heuristic techniques**: developed from the analysis of the performance of the standard gradient descent algorithm, e.g. **traingda**, **traingdx** and **trainrp**.
        2. **Numerical optimization techniques**: make use of the standard optimization techniques, e.g. **conjugate gradient** (**traincgf**, **traincgb**, **traincgp**, **trainscg**), **quasi-Newton** (**trainbfg**, **trainoss**), and Levenberg-Marquardt (**trainlm**).

Comparison of Training Algorithms

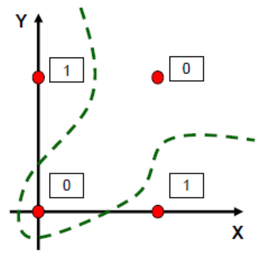
Training Algorithms		Comments
traingd	Gradient Descent (GD)	Original but slowest
traingdm	GD with momentum	Faster than traingd
traingda	GD with adaptive α	Faster than traingd, but can use for batch mode only.
traingdx	GD with adaptive α and with momentum	
trainrp	Resilient Backpropagation	Fast convergence
traincgf	Fletcher-Reeves Update	Conjugate Gradient Algorithms with fast convergence
traincgp	Polak-Ribiére Update	
traincgb	Powell-Beale Restarts	
trainscg	Scaled Conjugate Gradient	
trainbfg	BFGS algorithm	Quasi-Newton Algorithms with fast convergence
trainoss	One Step Secant algorithm	
trainlm	Levenberg-Marquardt	Fastest training. Memory reduction features
trainbr	Bayesian regularization	Improve generalization capability

Modeling Logical XOR Function

- The ***XOR solving problem*** using a simple backpropagation network

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

p_1	p_2	a
0	0	0
0	1	0
1	0	0
1	1	1




```
%Solution:
% Define the training inputs and targets
p = [0 0 1 1; 0 1 0 1];
t = [0 0 0 1];
% Create the backpropagation network
net = newff(minmax(p), [4 1], {'logsig', 'logsig'}, 'traingdx');
% Train the backpropagation network
net.trainParam.epochs = 500; % training stops if epochs reached
net.trainParam.show = 1; % plot the performance function at every epoch
net = train(net, p, t);
% Testing the performance of the trained backpropagation network
a = sim(net, p)
>> a = 0.0002     0.0011     0.0001     0.9985
>> t = 0          0          0          1
```