Neural Network basics and its Application

Mukesh A. Zaveri
Computer Engineering Department, SVNIT, Surat – 395007
mazaveri@coed.svnit.ac.in

Six days Workshop on Machine Learning – An approach to achieve Artificial Intelligence and its applications
Organized by Department of Information Technology
Chandubhai S. Patel Institute of Technology (CHARUSAT)
In association with GUJCOST and ACM student chapter CHARUSAT
Changa, Gujarat, during 18-23 December 2017

Soft Computing

- What is Soft Computing?
- Hard Computing Vs. Soft Computing
- Soft Computing Methods
  - Neural Network
  - Support Vector Machine
  - Genetic Algorithm
  - Fuzzy Logic
  - Probabilistic reasoning
- How to integrate / use for specific Application?
- Any other technique to be used with Soft Computing technique?

Soft Computing Techniques

- How to select soft computing technique?
- Neural networks –
  - learning, classification, optimization
- Fuzzy logic –
  - forming imprecision and reasoning on a semantic or linguistic level
- Genetic algorithm –
  - exploring set of all possible solutions
- Probabilistic reasoning
  - deals with uncertainty
- substantial areas of overlapping among different techniques
- they are complementary rather than competitive

Outline

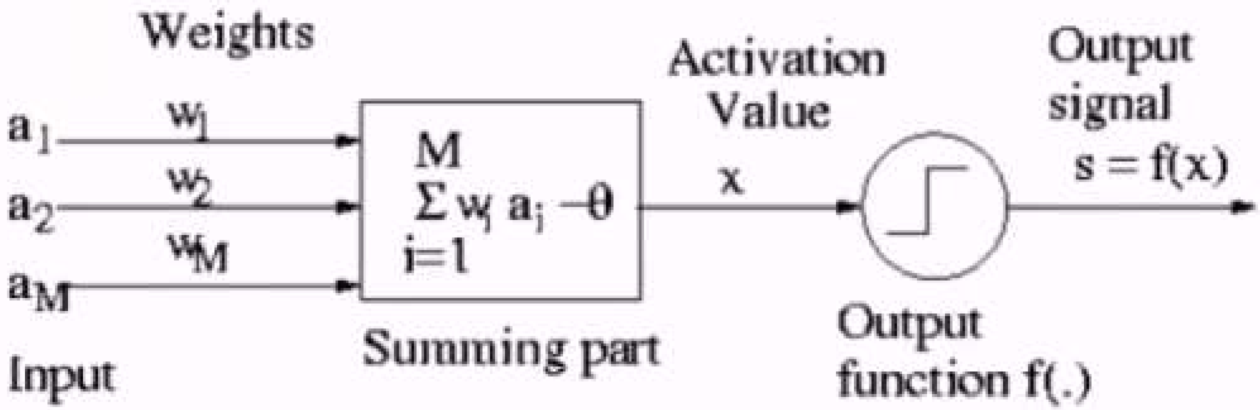
- Why Optimization?
- Problem Statement
- Neural Network
- Tracking using Neural Network
  - Problem Formulation
  - Proposed Algorithm
- Tracking using Genetic Algorithm
  - Problem Formulation
  - Proposed Algorithm
- Discussion

Why optimization?

- To achieve the goal with
  - Minimum computations
  - Minimum search
  - Efficient and Fast enough (for real time application)
  - At par with ideal goal (optimal solution)
- Optimization Techniques
  - Neural Network
  - Genetic Algorithm

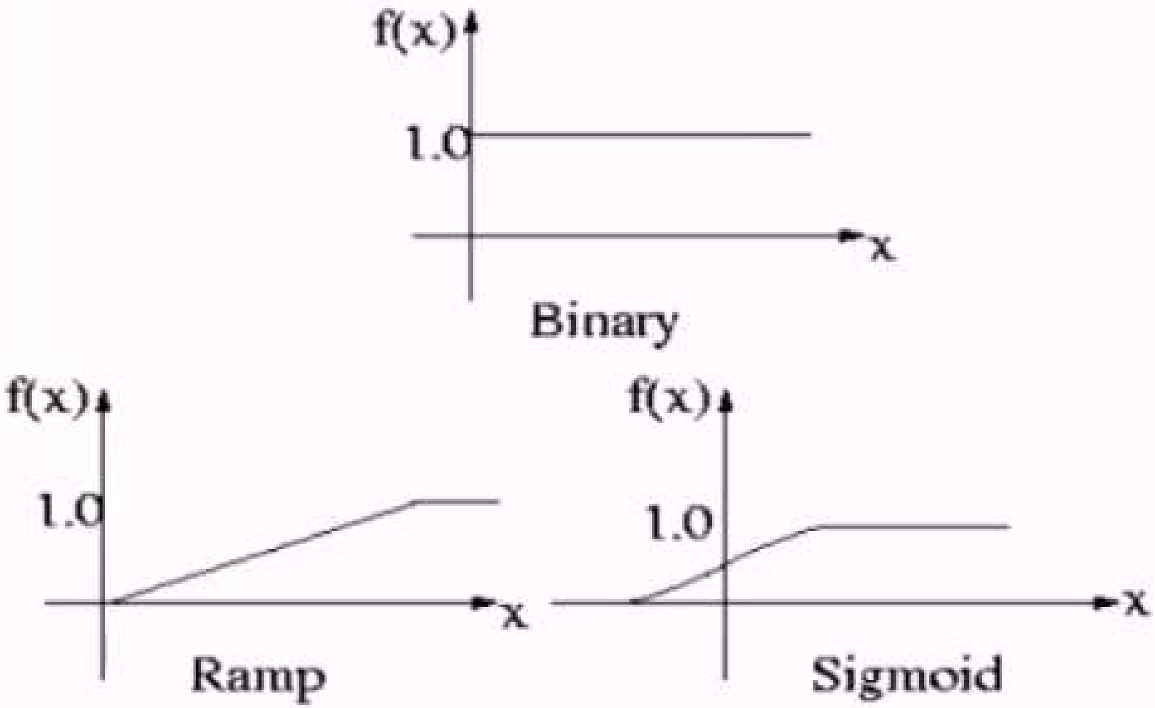
Introduction to Neural Network

Neuron Model



(Figure Src: ANN book)

Output Function



Output function (1) Logistic (2) Hyperbolic tangent

$$f(x) = \frac{1}{1 + \exp(-2\beta x)} \quad 0 \le f(x) \le 1$$

$$f(x) = \tanh(x) \quad -1 \le f(x) \le 1$$

(Figure Src: ANN book)

Classical Neuron Models

- McCulloch-Pitts model
  - Fixed weights, not capable of learning
- Rosenblatt's Perceptron model
  - Adjustment of weights

$$
\begin{array}{ll}
\text{Activation:} & x = \sum_{i=1}^{M} w_i a_i - \theta \\
\text{Output signal:} & s = f(x) \\
\text{Error:} & \delta = b - s \\
\text{Weight change:} & \Delta w_i = \eta \delta a_i
\end{array}
$$

Activation Dynamics

- Activation state: set of activation values at given instant
- Activation state space, Short term memory
- Activation dynamics leads to solution state
  - First order time derivative of activation state
- Stability: Equilibrium behavior of activation state
- Synchronous, asynchronous update

Activation models

Additive activation model

Dynamics equation: $\dot{x}_i(t) = -A_i x_i(t)$	Solution: $x_i(t) = x_i(0) \exp^{-A_i t}$
with nonzero resting potential	$\dot{x}_i(t) = -A_i x_i(t) + P_i$
Solution:	$x_i(t) = x_i(0) \exp^{-A_i t} + \frac{P_i}{A_i}(1 - \exp^{-A_i t})$
with constant external excitatory input	$\dot{x}_i(t) = -A_i x_i(t) + B_i I_i$
Solution:	$x_i(t) = x_i(0) \exp^{-A_i t} + \frac{B_i I_i}{A_i}(1 - \exp^{-A_i t})$

with external input and input from the output of the units:

(additive autoassociative model) $\quad \dot{x}_i(t) = -A_i x_i(t) + \sum_{j=1}^{N} f_j(x_j(t)) w_{ij} + B_i I_i$

Synaptic Dynamics

- Weight space, Long term memory
- Search for optimal weights for equilibrium state
  - Criteria: mean squared error, gradient descent, maximum likelihood, relative entropy
- Synaptic dynamics: trajectory of weight states
  - First order time derivative of weight state
- Learning algorithm
  - Supervised and unsupervised
- Convergence: Adjustment behavior of weights during learning

Learning Methods

Learning Function

$$\dot{\mathbf{w}}_i(t) = \eta g(\mathbf{w}_i(t), \mathbf{a}(t), b_i(t))\mathbf{a}(t)$$

$\mathbf{w}_i$	$(w_{i1}, w_{i2}, \ldots, w_{iM})^T$	weight vector for the ith unit
$\mathbf{a}$	$(a_1, a_2, \ldots, a_M)^T$	input vector
$\mathbf{b}$	$(b_1, b_2, \ldots, b_N)^T$	output vector
	for small increment	$\Delta\mathbf{w}_i(t) = \eta g(\mathbf{w}_i(t), \mathbf{a}(t), b_i(t))\mathbf{a}(t)$
		$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \Delta\mathbf{w}_i(t)$

Learning Methods

Learning Method	Synaptic dynamics equation
Hebbian	$\dot{w}_{ij}(t) = -w_{ij}(t) + s_i s_j$
Differential Hebbian	$\dot{w}_{ij}(t) = -w_{ij}(t) + s_i s_j + \dot{s}_i \dot{s}_j$
Competitive	$\dot{w}_{ij}(t) = -s_i w_{ij}(t) + s_i s_j$
Differential Competitive	$\dot{w}_{ij}(t) = \dot{s}_i [s_j - w_{ij}(t)]$
Error Correction	
Rosenblatt's perceptron	$\dot{w}_{ij}(t) = \eta(b_i - s_i)a_j \quad s_i = \mathrm{sgn}(\sum_j w_{ij} a_j)$
Continuous perceptron	$\dot{w}_{ij}(t) = \eta(b_i - s_i)\dot{f}_i(x_i)a_j$
(Delta learning)	$s_i = f_i(x_i)$ and $x_i = \sum_{j=1}^{M} w_{ij} a_j$

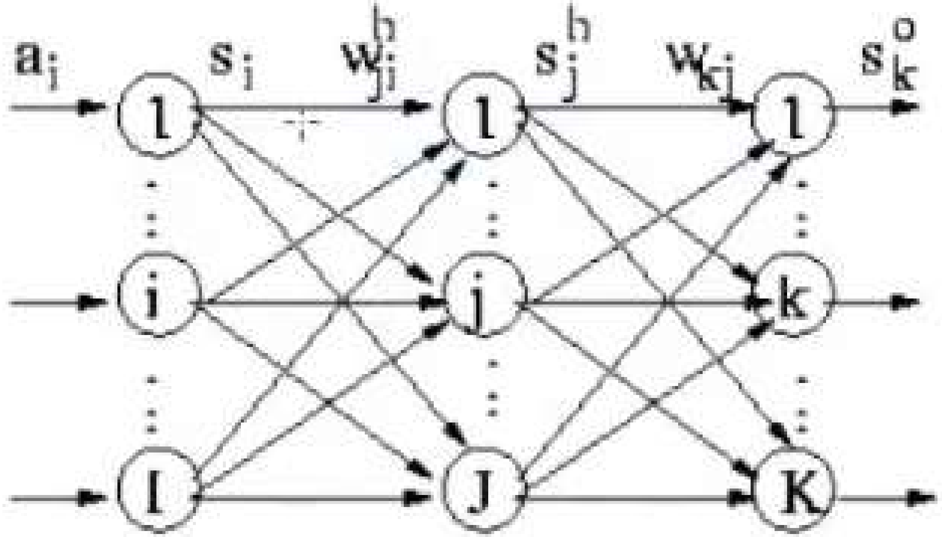
Learning Methods

Learning law	Weight adjustment	Initial weights	Learning
Hebbian	$\Delta w_{ij} = \eta f(\mathbf{w}_i^T \mathbf{a}) a_j = \eta s_i a_j$ for $j = 1, 2, \ldots, j$	0	Unsupervised
Perceptron	$\Delta w_{ij} = \eta [b_i - \operatorname{sgn}(\mathbf{w}_i^T \mathbf{a})] a_j = \eta (b_i - s_i) a_j$ for $j = 1, 2, \ldots, M$	Random	Supervised
Delta	$\Delta w_{ij} = \eta [b_i - f(\mathbf{w}_i^T \mathbf{a})] \dot{f}(\mathbf{w}_i^T \mathbf{a}) a_j = \eta [b_i - \mathbf{s}_i] \dot{f}(x_i) a_j$ for $j = 1, 2, \ldots, M$	Random	Supervised
Widrow-Hoff	$\Delta w_{ij} = \eta [b_i - \mathbf{w}_i^T \mathbf{a}] a_j$ for $j = 1, 2, \ldots, j$	Random	Supervised

[SRC: Zurada, 1992]

Feed forward and Feedback Neural Network

- Multilayer Perceptron Network
  - Generalized delta rule
- Hopfield Network
  - Bounded function and Symmetric weights



(Figure Src: ANN book)

Backpropagation Algorithm

A set of input-output patterns $(\mathbf{a}_l, \mathbf{b}_l)$, $l = 1, 2, \ldots, L$
lth input vector $\mathbf{a}_l = (a_{l1}, a_{l2}, \ldots, a_{lI})^T$
lth output vector $\mathbf{b}_l = (b_{l1}, b_{l2}, \ldots, b_{lK})^T$
Let $\mathbf{a} = \mathbf{a}(m) = \mathbf{a}_l$ and $\mathbf{b} = \mathbf{b}(m) = \mathbf{b}_l$

Activation of unit i in the input layer	$x_i = a_i(m)$
Activation of unit j in the hidden layer	$x_j^h = \sum_{i=1}^{I} w_{ji}^h x_i$
Output signal from jth unit in the hidden layer	$s_j^h = f_j^h(x_j^h)$
Activation of unit k in the output layer	$x_k^o = \sum_{j=1}^{J} w_{kj} s_j^h$
Output signal from unit k in the output layer	$s_k^o = f_k^o(x_k^o)$
Error term for the kth output unit	$\delta_k^o = (b_k - s_k^o)\dot{f}_k^o$
Update weights on output layer	$w_{kj}(m+1) = w_{kj}(m) + \eta\delta_k^o s_j^h$
Error term for the jth hidden unit	$\delta_j^h = \dot{f}_j^h \sum_{k=1}^{K} \delta_k^o w_{kj}$
Update the weights on the hidden layer	$w_{ji}^h(m+1) = w_{ji}^h(m) + \eta\delta_j^h a_i$
Calculate the error for the lth pattern	$E_l = \frac{1}{2}\sum_{k=1}^{K}(b_{lk} - s_k^o)^2$
Total error for all patterns	$E = \sum_{l=1}^{L} E_l$

Apply the given patterns one by one, several times in random order
Update the weights until the total error reduces to an acceptable value

Radial Basis Function Network

- Feed forward Network with one hidden layer
- Hidden layer unit has a receptive field
- Training: Deciding centre and sharpness of Gaussian
- Advantage: Extra units with centres near parts of input which are difficult to classify
  - Many sensory neurons respond only to some small subspace of input space and are silent in response to other inputs

$$f(x) = \sum_{i=1}^{r} w_i \phi_i(\|\mathbf{X} - \mu_i\|, \theta_i)$$

$\mathbf{X}$ input vector ϕ_i Basis function

w_i weights

$\mu_i = (\mu_{i1}, \mu_{i2}, \ldots, \mu_{in})^T$ center vector

$\theta_i = (\theta_{i1}, \theta_{i2}, \ldots, \theta_{in})^T$ bandwidth vector of ith node

if ϕ_i is Gaussian

$$\phi_i(\|\mathbf{X} - \mu_i\|, \theta_i) = \exp\{[-\|\mathbf{X} - \mu_i\|/(\theta_{i1}, \theta_{i2}, \ldots, \theta_{in})]^2\}$$

Support Vector Machine

- Pattern analysis, classification
- Linear separable, nonlinear separable problems
- When to use SVM?
  - Unknown and nonlinear dependency
  - High dimensional data Learning Methods
  - No information about underlying joint probability function
- Features
  - Perform distribution free learning
  - Nonparametric: parameters not predefined and their number depends on training data
  - Structural Risk Minimization (SRM)

Applications

- Investment analysis
- Signature analysis
- Process control and Monitoring
- Marketing
- Character recognition
- Information retrieval
- Travelling salesman problem
- Image pattern recognition, segmentation
- Vector quantization
- Texture classification and segmentation

Advanced Soft Computing Techniques (Incomplete List)

- Advanced Neural Network
  - Probabilistic Neural Network
  - Deep Belief Neural Network
  - Recurrent Neural Network
  - Convolution Neural Network
- Advanced Fuzzy based Techniques
  - Type II Fuzzy logic
  - Rough Set Theory
- Biological / Nature inspired Techniques
  - Ant Colony
  - Particle Swarm Optimization