Perceptron versus MLP

Introduction

- Garry Kasparov (may 1997) vs IBM supercomputer –deep blue (DB)
- DB – chess-playing programs
- DB – analyzing 200 million positions a second

Introduction

- Machine learning – enable computer to learn from experience, learn by example and learn by analogy
- The popular – ANN and GA
- What is NN?

Neural network

- Model of reasoning based on human brain
- Consists of a number of neurons/nodes
- Neurons are connected by weighted links passing signals from one neuron to another
- How does an NN learns?
  - Through repeated adjustments of weights

Neuron

- Basis of most NN
- Proposed by Warren McCulloch and Walter Pitts (1943)
- Use activation function called sign function
- Sign function?

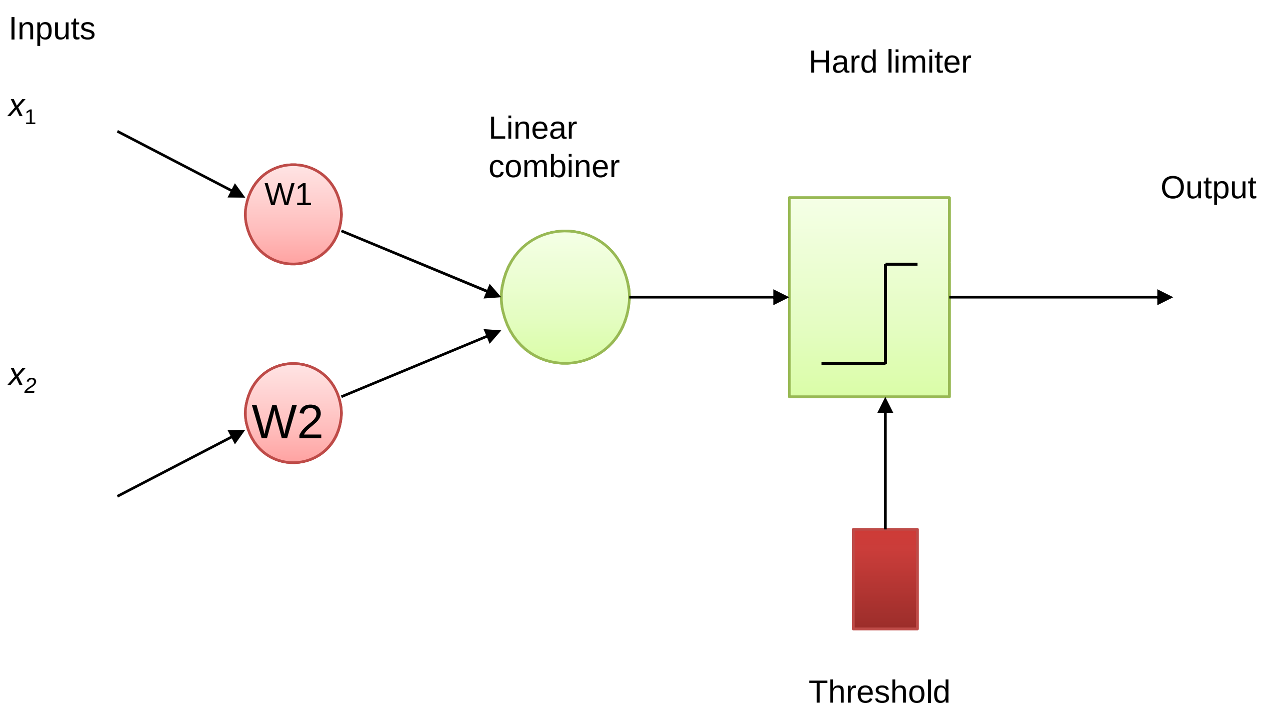
Neuron

- Four common choices of AF:
  1. step function-hard limit function for classification and pattern recognition
  2. sign function-hard limit function, for classification and pattern recognition
  3. sigmoid function-use for Backpropagation networks
  4. linear approximation function

Perceptron

- The simplest form of NN, consists of a single neuron with adjustable weights and a hard limiter function
- Perceptron learning rule proposed by Rosenblatt (1958). Learning rule? Learning algorithm?
- Perceptron is based on McCulloch and Pitts neuron model

Perceptron – single neuron



Perceptron – single neuron

Initialization of weights and threshold, calculate $Y(p)$ based on the selected activation function to determine $e(p)$ and then update the weights.

$$e(p) = Y_d(p) - Y(p)$$

Weight correction/updated is computed by delta rule

$$\Delta w_i(p) = \alpha \times x_i(p) \times e(p)$$

$$w_i(p+1) = w_i(p) + \Delta w_i(p)$$

***Small learning rate, small changes to the delta weight

Multilayer neural network

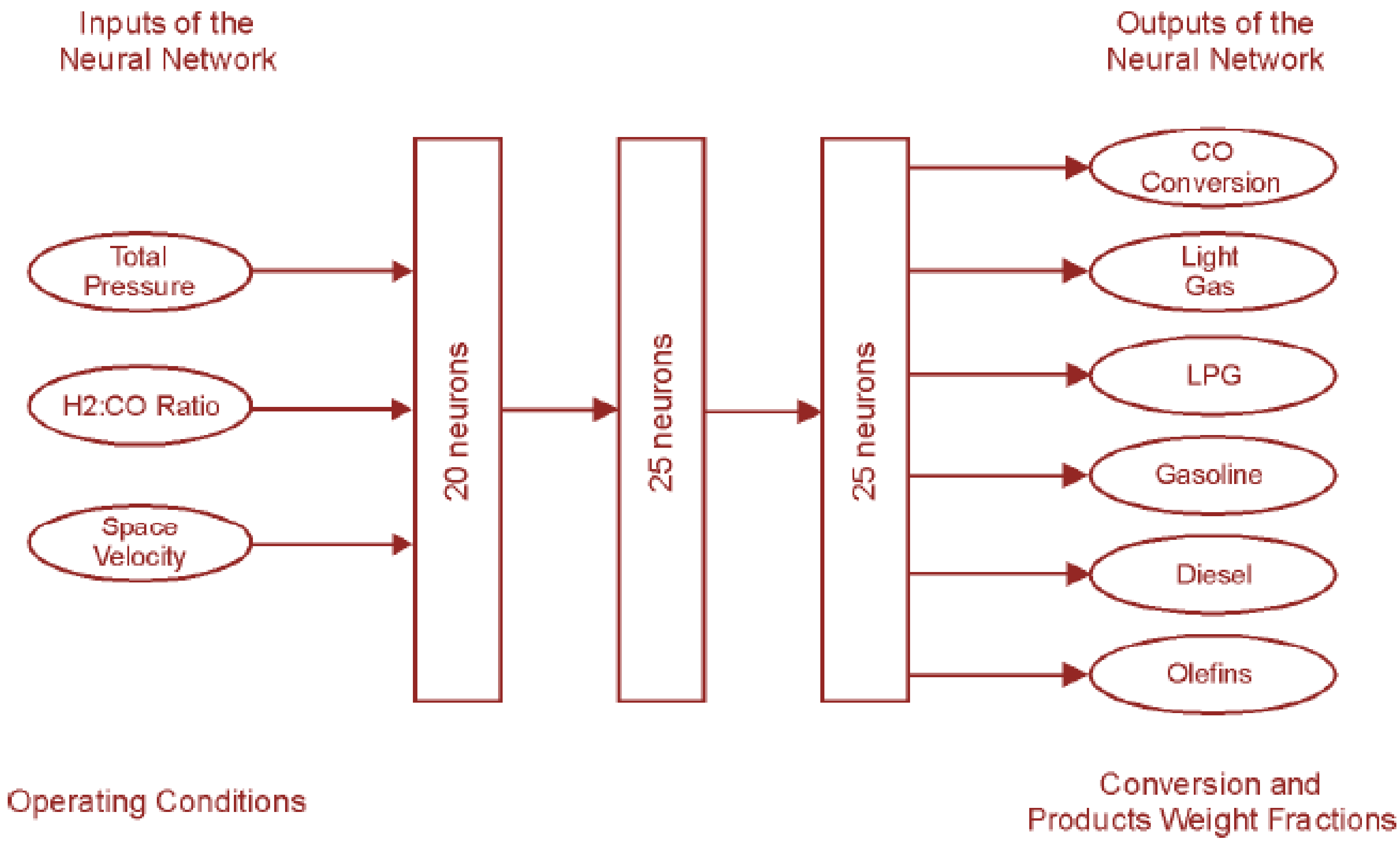


Figure 1. Neural network flowchart.

Multilayer neural network

i, j, k = notation for input, hidden, output layer respectively

Weight correction for output layer (*k*)

Output of hidden layer

$$\Delta w_{jk}(p) = \alpha \times y_j(p) \times \delta_k(p)$$

$$\delta_k(p) = error\ \ gradient\ \ at\ \ node\ \ k$$

$$\delta_k(p) = \frac{\partial y_k(p)}{\partial X_k(p)} \times e_k(p)$$

Multilayer neural network

If the function y uses is the sigmoid function, then

$$\delta_k(p) = \frac{\partial\left\{\dfrac{1}{1+\exp(-X_k(p))}\right\}}{\partial X_k(p)} \times e_k(p) = \frac{\exp[-X_k(p)]}{\{1+\exp[-X_k(p)]\}^2} \times e_k(p)$$

we obtain :

$$\delta_k(p) = y_k(p) \times [1 - y_k(p)] \times e_k(p)$$

where

$$y_k(p) = \frac{1}{1+\exp[-X_k(p)]}$$

Multilayer neural network

How about weight correction for the hidden layer?

$$\Delta w_{ij}(p)$$

$$\Delta w_{ij}(p) = \alpha \times x_i(p) \times \delta_j(p)$$

$$\delta_j(p) = y_j(p) \times [1 - y_j(p)] \times \sum_{k=1}^{l} \delta_k(p) w_{jk}(p)$$

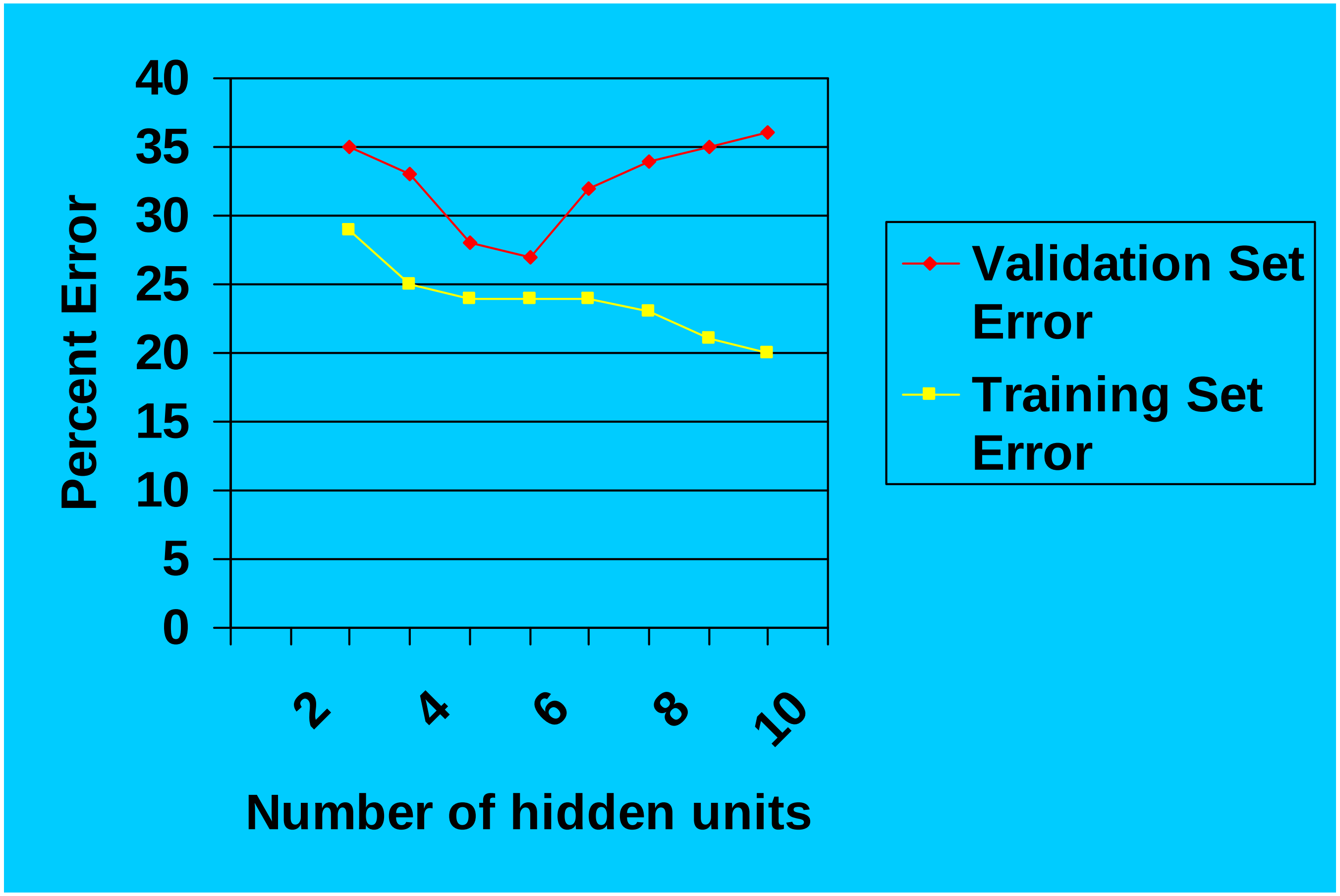
$$l = number \quad of \quad neurons \quad in \quad output \quad layer$$

$$y_j(p) = \frac{1}{1 + \exp[-X_j(p)]}$$

$$X_j(p) = \sum_{i=1}^{n} x_i(p) \times w_{ij}(p) - \theta_j$$

$$n = number \quad of \quad neurons \quad in \quad input \quad layer$$

ANN Design Balance: Depth



- Too few hidden layers will cause errors in accuracy
- Too many hidden layers will cause errors in generalization!

Neural Network Summaries

- In general, MLP with hyperbolic tangent learns faster than sigmoid activation function
- We can accelerate training by including a momentum term, and equation with that term is called generalized delta rule.
- Hopfield network is a recurrent network.
- Supervised learning is an active learning.
- Other name for unsupervised learning is self-organized learning and suitable for classification tasks. It learn much faster than BP networks.

Activity in Class

- Compare weight correction for perceptron and MLP. What are the differences?
- What is a useful indicator for network's performance?
- What is a major problem when using BP learning?

Activity in class

Construct 3 inputs and 1 output perceptron. Given the initial weights as 2, 3 and 4, learning rate of 0.1, determine the predicted output.