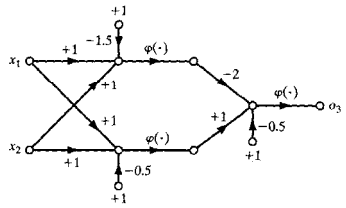
# Multilayer Perceptrons

The Essence of Neural Networks

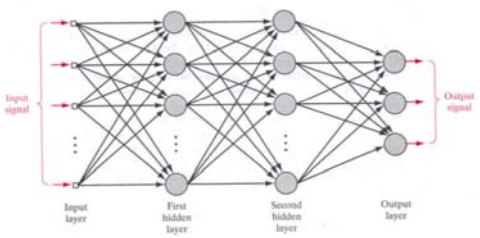
## XOR Problem



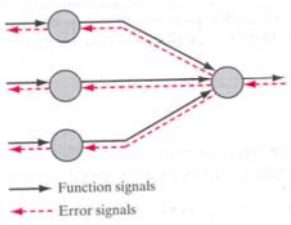

## Multilayer Perceptrons

- Several layers of interconnected neurons:
  - Sensory (input) layer
  - Output layer
  - One or more hidden layers
- Generalization of single layer perceptrons
- Relies on non-linear activation functions

## Multilayer Perceptrons




## Signal Flow




## Role of Hidden Perceptrons

- Hidden perceptrons act as feature extractors:
  - As the learning process proceeds, hidden neurons gradually discover the salient features of the problem space

## Credit Assignment

- Error signals are easily computable at the output level
- Each of the hidden layers contributed to the error signal
- Credit (or blame) needs to be assigned to each of the hidden neurons in order to adjust its weights accordingly to ensure proper operation of the entire system



## Error Signals

- Training Sample: $\mathcal{T} = \{\mathbf{x}(n), \mathbf{d}(n)\}_{n=1}^{N}$
- Error at each output node: $e_j(n) = d_j(n) - y_j(n)$
- Error energy at each output node: $\mathscr{E}_j(n) = \frac{1}{2} e_j^2(n)$
- Total error energy: $\mathscr{E}(n) = \sum_{j \in C} \mathscr{E}_j(n)$
  $= \frac{1}{2} \sum_{j \in C} e_j^2(n)$
- Average error energy: $\mathscr{E}_{av}(N) = \frac{1}{N} \sum_{n=1}^{N} \mathscr{E}(n)$
  $= \frac{1}{2N} \sum_{n=1}^{N} \sum_{j \in C} e_j^2(n)$



## Generalized Learning Algorithm

1. Apply all the data in the learning set
2. Adjust the synaptic weights
3. Reapply the data in the set

Repeat steps 1-3 until some measure of convergence is achieved

Each application of all the data in the learning set is called an “Epoch”

## Learning Methods

- Batch Learning
- Online Learning



## Batch Learning

- Synaptic weights are adjusted after each application of all the data in the learning set (i.e. after each epoch)
- The average error energy is used as the cost function
- Weight adjustment follows the steepest gradient
- Major advantage: easily parallelizable



## Online Learning

- Adjustment of the synaptic weights is performed after the presentation of each data sample
- The error energy is used as the cost function
- The data samples in the training set are randomly shuffled after each epoch
- Often referred to as stochastic learning

## Advantages / Disadvantages of Online Learning

- Advantages:
  - The stochastic nature of the learning process reduces the possibility of getting stuck in local minima
  - Easily takes advantage of redundant data
  - Easy to implement
- Disadvantage: Cannot be parallelized



## Back Propagation Algorithm

- A method of online learning
- Two passes:
  - Forward pass to compute the outputs and the error signals
  - Backward pass to “propagate” the error signals to adjust synaptic weights
- Generalization of error correction learning and LMS algorithm



## Forward Pass

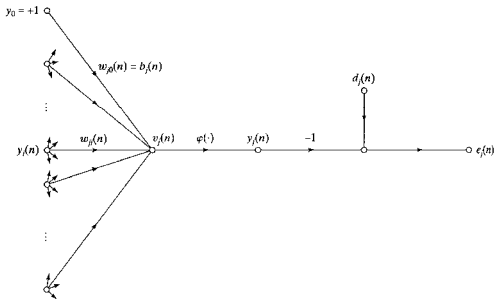
- Starting at the input layer, use the inputs from the training set to compute the outputs from this layer
- Iteratively use the outputs from each layer as inputs to the next layer to compute the outputs from that layer
- When the output layer is reached, compute the error signal

## Backward Pass

- Starting at the output layer:
  - Use the error signal to compute the gradient
  - Use the gradient to compute the change in the weights for this layer
- Going backward one step at a time:
  - Compute the gradients at successive layers
  - Use the gradients to compute the change in the weights for this layer
- This process constitutes a form of credit assignment



## Signal Flow






## Weight Adjustment

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n)$$

$$y_j(n) = \varphi_j(v_j(n))$$

$$\mathscr{E}_j(n) = \frac{1}{2} e_j^2(n)$$

$$\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathscr{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

$$\frac{\partial \mathscr{E}(n)}{\partial e_j(n)} = e_j(n) \qquad \frac{\partial e_j(n)}{\partial y_j(n)} = -1 \qquad \frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n)) \qquad \frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n)$$

$$\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} = -e_j(n) \varphi_j'(v_j(n)) y_i(n)$$

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)}$$

$$= \eta e_j(n) \varphi_j'(v_j(n)) y_i(n)$$

## Error Correction

$$\Delta w_{ji}(n) = \eta e_j(n)\varphi_j'(v_j(n))y_i(n)$$

- For output neurons: apply above formula
- For hidden neurons: need to define error signal

## Back Propagation

- For output nodes, define the local gradient:

$$\delta_j(n) = e_j(n)\varphi_j'(v_j(n))$$

- For hidden nodes, the local gradient can be computed as (see proof):

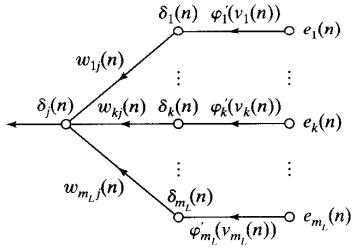
$$\delta_j(n) = \varphi_j'(v_j(n))\sum_k \delta_k(n)w_{kj}(n)$$

- All nodes:

$$\Delta w_{ji}(n) = \eta\delta_j(n)y_i(n)$$

## Back Propagation

- Error signals are propagated back one layer at a time:



## Activation Function

- Activation functions are typically non-linear
- A multilayer perceptron with a linear activation function is equivalent to a single layer perceptron
- The activation function is typically chosen to be a sigmoid function (biologically inspired) of the form:

$$y_j = \frac{1}{1+\exp(-av_j)} \; or:$$

$$y_j = a\tanh(bv_j(n))$$



## Function Differentiation

$$\varphi_j(v_j(n)) = \frac{1}{1+\exp(-av_j(n))}$$

$$\varphi_j'(v_j(n)) = \frac{a\exp(-av_j(n))}{[1+\exp(-av_j(n))]^2}$$

$$\varphi_j'(v_j(n)) = a\frac{[1+\exp(-av_j(n))-1]}{[1+\exp(-av_j(n))]^2}$$

$$\varphi_j'(v_j(n)) = a\left[\frac{1+\exp(-av_j(n))}{[1+\exp(-av_j(n))]^2} - \frac{1}{[1+\exp(-av_j(n))]^2}\right]$$

$$\varphi_j'(v_j(n)) = a[y(n) - y^2(n)] = ay(n)[1-y(n)]$$



## Function Differentiation

$$\varphi_j(v_j(n)) = a\tanh(bv_j(n))$$

$$\varphi_j'(v_j(n)) = ab\sec h^2(bv_j(n))$$

$$\varphi_j'(v_j(n)) = ab(1-\tanh^2(bv_j n))$$

$$\varphi_j'(v_j(n)) = \frac{b}{a}(a^2 - a^2\tanh^2(bv_j n))$$

$$\varphi_j'(v_j(n)) = \frac{b}{a}(a - a\tanh(bv_j(n))(a + a\tanh(bv_j(n))$$

$$\varphi_j'(v_j(n)) = \frac{b}{a}(a - y_j(n))(a + y_j(n))$$

## Learning Rate Modification

- The learning rate may be constant across the network or may be connection-dependent
- The learning rate may be constant across iterations or may be altered for initial fast convergence followed by fine tuning
- A momentum factor may be introduced to avoid oscillatory behavior:

$$\Delta w_{ij}(n) = \alpha \Delta w_{ij}(n-1) + \eta \delta_j(n) y_i(n)$$



## Training Process Revisited

- Assuming a set of training data of size N
- Present the data to the network and adjust the weights after every data item
- After the full training set is used, more iterations may be necessary
- A full iteration is called an *epoch*
- It is prudent to randomize the order of the data items between epochs



## Stopping Criteria

- When rate of change of the weights is less than a given threshold
- When the performance of the network over new data is satisfactory
- When the change in performance is less than a given threshold

## Performance Heuristics

- Use sequential (online) processing
- Carefully select the training examples to cover as much as possible of the application domain
- Antisymmetric activation functions converge faster than non-symmetric functions (Popular function: a tanh(bh) where a=1.7159, b=2/3)

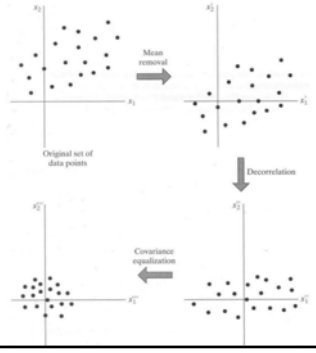


## Performance Heuristics

- Make sure that the target values fall within the non-saturated range of the activation function
- Normalize the input variables (mean removal, de-correlation, covariance equalization)



## Normalization



## Performance Heuristics

- Select the initial weights to have the outputs fall mid-range between origin and saturation
- Learning from hints (use knowledge of the system in the learning process)



## Performance Heuristics

- Selection of learning Rate:
  - Later layers should have a smaller learning rate than earlier layers
  - Neurons with more inputs should have a fewer learning rate than neurons with more inputs



## Jacobian Matrix and Convergence

- The Jacobian matrix of derivatives is composed of the partial derivates with respect to the weights
- Each row represents a training data point
- In most cases the matrix is ill conditioned (in some cases rank-deficient)
- A rank deficient matrix leads to very slow conversion

## Multiple Output Neural Networks

- Can a neural network with m outputs be treated as m independent single-output neural networks?
- Why?



## Effect of Training Set Size

- In practice, the larger the training set, the smaller the number of epochs required to obtain the same average error



## Feature Space

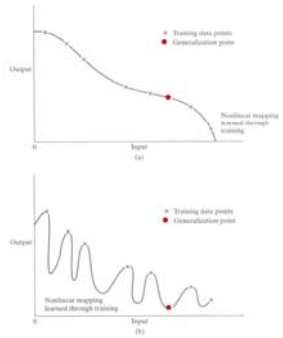
- The outputs from the hidden layers are collectively know as the “hidden space”
- After training, the hidden space can be viewed as a feature space describing the salient features of the represented domain

## Generalization

- A neural network is said to correctly generalize if it produces correct (or approximately correct) results given previously unseen input
- Proper generalization depends on the training set



## Generalization






## Over Training

- With a very large training set, a network can be over trained:
  - A previously seen input will produce a very accurate output
  - A previously unseen input will produce an incorrect output

## Generalization

- Empirically, proper generalization is accomplished with a data set of size:

$$N = O\left(\frac{W}{\varepsilon}\right)$$

where W = the number of synapses, ε = the desired error tolerance



## Universal Approximation Theorem

- Any function $f(x_1, ..., x_{m_0})$ can be approximated as a summation of a family of non-constant, bounded, monotonically-increasing continuous functions:

$$F(x_1, ..., x_{m_0}) = \sum_{i=1}^{m_1} \alpha_i \varphi\left(\sum_{j=1}^{m_0} w_{ij} x_j + b_i\right)$$

such that:

$$|F(x_1, ..., x_{m_0}) - f(x_1, ..., x_{m_0})| < \varepsilon$$

*for all $x_1, x_2, ..., x_{m_0}$ that lie in the input space.*



## Conflicting Requirements

- A large number of hidden nodes is necessary for best approximation
- The ratio of the number of hidden nodes to the size of the training sample should be small for a better fit to the approximation
- Using results from approximation by Fourier series, we need a sample of size:

$$N = \frac{m_0 m_1}{\varepsilon_0}$$

Where $m_0$ = number of inputs, $m_1$ = number of hidden nodes, $\varepsilon_0$ = acceptable mean squared error

## Function Approximation

- A single hidden layer is generally sufficient
- By using multiple hidden layers, the rate of convergence is generally enhanced
- It is a common practice to use 2 hidden layers
- The training set should be representative of the function

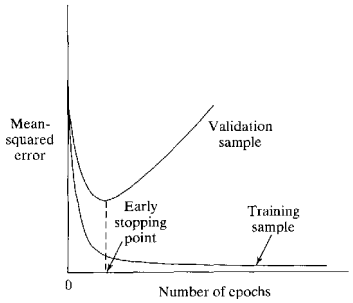


## Cross Validation

- The training data set is divided into 2 subsets:
  - Estimation subset
  - Validation subset
- Training steps:
  - Parameterize the system using the estimation data set
  - Test the resulting system using the validation subset
- Typical split:
  - 80 - 95% of the data set for estimation
  - 5 - 20% of the data set for validation



## Early Stopping of Training



## Early Stopping of Training

- Apply training using the estimation data set
- Stop the training every fixed number of epochs
- Check the accuracy of the network using the validation data set
- Stop the training when the error in the validation set starts to increase
- Potential problem: local minima



## Multifold Cross Validation

- Divide the training set into k subsets
- For each epoch:
  - Use the entries from all subsets except one as an estimation set
  - Use the remaining subset as a validation set
- Use a different subset for validation for each epoch
- Useful for small training sets



## Effect of Network Size

- An undersized network risks being insufficient to represent the required behavior
- An oversized network risks over-training
- To optimize the size of the network, we use one of 2 techniques:
  - Network growing
  - Network pruning

## Network Pruning

- During training the network is examined and synapses with sufficiently small weights are eliminated
- Several statistical metrics are available for determining which weights are sufficiently small
- Re-training is usually needed after pruning



## Computational Complexity of Back propagation

- What is the computational complexity of back propagation?



## Benefits of Multilayer Perceptrons

- Connectionist: used as a metaphor for biological neural networks
- Computationally efficient
  - Can easily be parallelized
- Universal computing machines

## Drawbacks of Multilayer Perceptrons

- Convergence can be slow
- Local minima can affect the training process
- Hard to scale

52

## Multi-Layer NN as Pattern Replicator

- Output = Input
- Useful for filtering operations
- Can we use back propagation? What plays the role of the teacher?
- How about pattern identification?

53

## Convergence Acceleration Heuristics

- Assign different learning rates to different synaptic weights
- Vary the learning rates for different iterations
- Increase the learning rate if the derivative (of the cost function) has the same algebraic sign for several consecutive iterations
- Decrease the learning rate if the derivative (of the cost function) alternates algebraic signs for several consecutive iterations

54

## Other Learning Techniques

- Other learning techniques based on optimization techniques exist, e.g.:
  - Conjugate-gradient method
  - Quasi-Newton method
- These methods are harder to use
- These methods are particularly useful when training data is limited

55

## Convolutional Networks

- A biologically inspired technique where a network is structured of successive networks that perform specialized functions such as:
  - Feature extraction
  - Subsampling
  - Scaling
  - Shifting
- Particularly suited for recognition of two dimensional shapes

56

## Non Linear Filtering

- Also known as temporal pattern recognition
- Refers to situations where the output depends on the current and historic inputs
- Can be accomplished using time delay modules whose output is treated as additional input

57

## Small Scale vs. Large Scale Learning

- Small scale learning problems are problems where the learning process is limited by the size of the available training data set
- Large scale learning problems are problems where the learning process is limited by the available computation time
- Both will have an effect on the accuracy of the trained network