Neural Networks for Forecasting and Optimization: The Untapped Potential

KEN OTWELL
DIRECTOR OF RESEARCH
BEHAVHEURISTICS, INC.
KOTWELL@DELPHI.COM

Presentation Abstract

Neural networks are the latest variety of computational systems designed to produce, or at least mimic, intelligent behavior. Unlike 'classical' artificial intelligence systems which are designed to directly emulate rational, logical reasoning; neural network researchers are trying to reproduce the underlying processing mechanisms that give rise to intelligence as an emergent property of complex, adaptive systems. Neural network systems have been developed in the laboratory for pattern recognition, forecasting, motor control, and even some forms of intuitive problem solving. The most widespread component of these systems is the 'backpropagation' algorithm for assigning and distributing error to each of a large number of adaptive components. Just as the immense variety of artificial intelligence research produced in the laboratory has been filtered down to "expert systems'" technology by the market place, virtually all commercial applications of neural network technology are based on backpropagation of error.

Commercial neural networks are quickly becoming established as indispensable tools in the financial analyst's arsenal. They are now used in a wide variety of roles, from forecasting price changes of both equities and bonds to predicting market shifts between value-based and speculative modes. Many users of these systems are quietly (and some not so quietly) making money by trading based on the output of these systems. However, there are fundamentally more sophisticated techniques emerging from the laboratory, for both forecasting and direct portfolio optimization, which have yet to be embraced in commercial applications.

This presentation is designed to familiarize the audience with core neural network technology, discuss how it is now being used in the financial industry, and provide insights on how it can be, and will be, used more effectively in the very near future. We will first discuss the fundamental algorithms of neural networks and illustrate their strong relationship to econometric techniques. We will next review some of the applications that are now trading and discuss how the technology is being applied. Finally, we will discuss advanced techniques, focusing on time-lagged and simultaneous recurrent networks and their use in multi-network architectures for optimization over time in the form of approximate dynamic programming.

Introduction

This paper consists primarily of the presentation text and graphics from the talk at the Northfield conference. They have been edited for clarity and flow. Additional commentary has been added for reinforcing the key ideas.

Current Applications

Forecasting...

- Stocks
- Commodity Futures
- Market Indexes
- Exchange Rates
- Economic Turning Points
- Bond Ratings
- Loan Risks
- Bankruptcies

Pattern Recognition...

- Credit Card Fraud Detection

Decision Support...

- Portfolio Management
- Asset Allocation

Most financial applications of neural networks are strictly in the area of forecasting. This requires a large history of data for "training" the network to achieve useful results. Naive extrapolation of a trained neural network into novel market modes where no data exists is risky.

Future Trading Style Enabled by Advanced Technology:

Predictions

Applications will migrate:

- from error-based *forecasting*
  (integration of diverse indicators)

- to utility-based *optimization*
  (creatively meeting client objectives)

Primary analyst functions will become:

- Problem Definition
- Utility Function Development
- Data Selection and Knowledge Insertion
- ***Not Model Building***

There are virtually no current applications which take the next logical step up from forecasting and actually make decisions as to when and how much to buy or sell, or what percentage of a portfolio to allocate to which equity, bond, or market. Additionally, the technology for optimally using market feedback and effects of "self-generated" trades has been developed in other industries (primarily chemical process control) and is ready for transference. This is a very large area of untapped potential.

Basic Adaptive Networks

Neural Network Definition

Artificial Neural Networks are computational mechanisms (hardware and/or software) to adaptively learn and compute arbitrary, non-linear *functional mappings* between data sets, or to *adapt optimal responses*, using *local processing*.

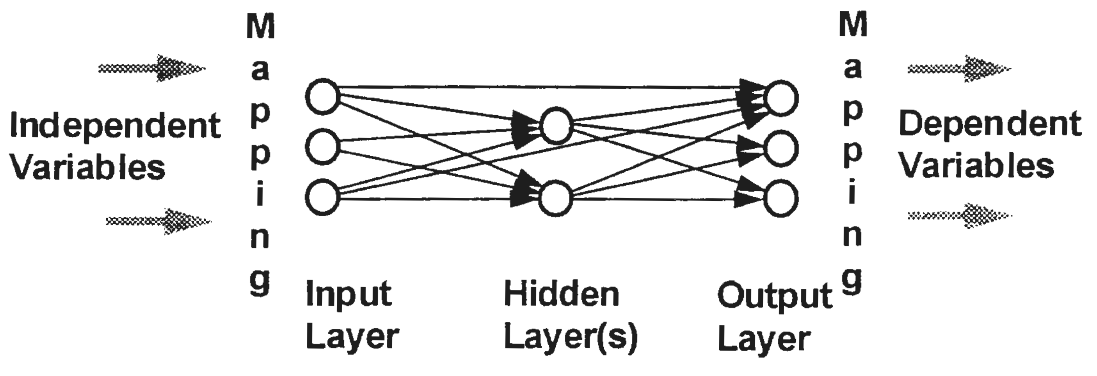
Functional mappings can be Boolean, fuzzy, continuous, stochastic, dynamic, probabilistic or a hybrid.

Optimal responses can be bootstrapped from evaluative feedback without requiring prior knowledge of best response.

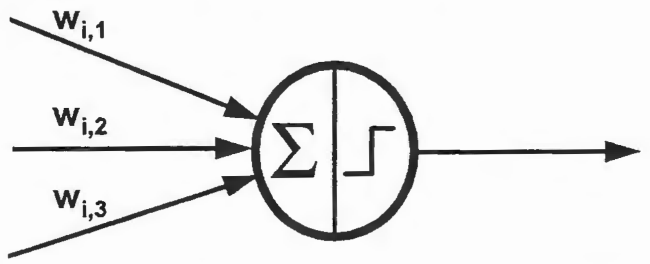
Local processing can be highly parallelized for speed.

Feedforward Neural Network Architecture

○	**Node**	**- Short-term Memory: Activation State** **- Computes output Transformation**
→	**Connection**	**- Long-Term Memory: Connection Weight** **- Computes Activation Transfer**



McCulloch and Pitts Neuron



$$\text{input}_i(t+1) = \Sigma\ \text{weight}_{i,j}\ \text{output}_j(t)$$
$$\text{output}_i(t+1) = \Theta(\ \text{input}_i(t+1) - \text{threshold})$$

$$\Theta(x) = \begin{cases} 1 \text{ if } x \geq 0; \\ 0 \text{ otherwise.} \end{cases}$$

- First Neuron Model: McCulloch and Pitts (1943)
- Binary Threshold Unit
- Step Function or Heaviside Function
- Proved Universal Computing Ability

The structure of the basic neuron model has not changed since it was initially developed by McCulloch and Pitts. Virtually all commercial neural networks today use a separate input and output function of some form. Since any logic gate can be constructed from this model by appropriate selection of the weighting parameters, a network of such nodes can emulate any computer.

Neural Network Paradigms

Network Architectures

- Feedforward: A signal only flows through each processing element once per "firing."
- Recurrent: Signals may flow through each processing elements many times prior to generating output.

Node Dynamics

- Deterministic: Node output is uniquely determined from its input.
- Stochastic: Node output is stochastically selected from a distribution parameterized by its input.

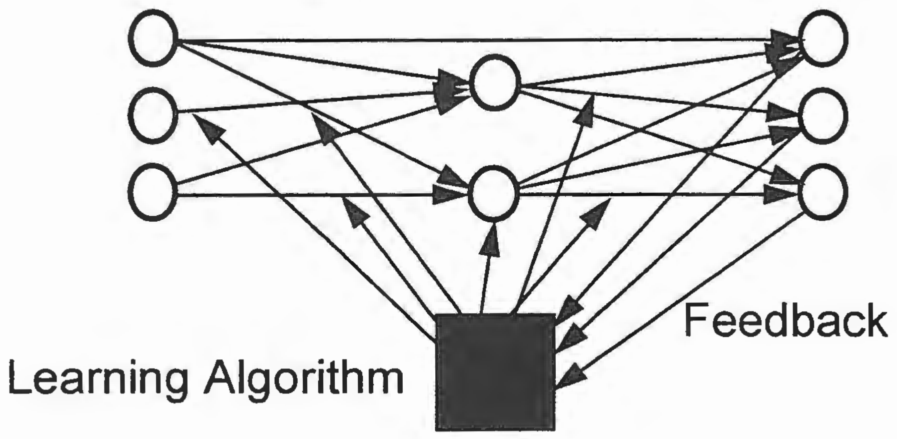
Connection Dynamics

- Synchronous: Clock-controlled Dataflow.
- Asynchronous: Probabilistic Dataflow

The vast majority of current applications use deterministic, feedforward architectures which are trained by the backpropagation of error algorithm. Asynchronous stochastic networks show great promise and have received a lot of publicity. They are being heavily researched (e.g., at the Sante Fe Institute) but have not yet been developed to the point where they can outperform current techniques on existing hardware.

However, deterministic recurrent systems are now showing tremendous improvement over classical techniques in areas such as optimal control of chemical processes and adaptive avionics control systems. They will form the next marketable wave of neural network technology.

Learning in Neural Networks



- Feedback:
- Adapts Network Configuration (Mathematical Model)
- Adapts Connection Weights (Model Parameters)

Supervised Training - Requires a Teacher

- Quantified Feedback
  - » Error Function
  - » Minimum Loss Function
  - » Utility Function
- Reinforcement
  - » Directional: Higher, Lower

Evaluative/Critic: Good, Bad

Unsupervised Training - Requires Redundancy

- Representational Transformation
  - »Feature Extraction/Enhancement
  - »Data/Dimensionality Reduction

Competitive Learning (Winner-take-all)
Categorization/Clustering

Hebbian Learning

When a neuron stimulates another neuron while the second neuron is firing, the connection from the first to the second is proportionally strengthened. (Donald Hebb, 1949). Neo-Hebbian dynamical system for modeling classical (Pavlovian) conditioning:

$$\text{input}_j(t+1) = \Sigma_i \text{weight}_{i,j}(t) \times \text{output}_j(t) + \text{input}_j(t) - \text{decay} \times \text{output}_j(t)$$

$$\text{output}_j(t+1) = \max(0, \text{input}_j(t+1) - \text{threshold})$$

$$\text{weightChange}_{i,j}(t+1) = \text{learningRate} \times \text{input}_j(t) \times \text{output}_i(t) - \text{decay} \times \text{weight}_{i,j}(t)$$

Linear, Feedforward, Deterministic Networks

Node Activation

$\text{input}_i(t+1) = \Sigma\ \text{weight}_{i,j}\ \text{output}_j(t)$

$\text{output}_i(t+1) = \text{input}_i(t+1)$ (no transformation)

- Adaline (Widrow & Hoff, 1960)
- Requires Linear Independence in Patterns

$\text{output}_i(t+1) = \Theta(\text{input}_i(t+1) - \text{threshold})$ (Step Function)

- Perceptron (Rosenblatt, 1962)
- Requires Linear Separability of Patterns

Weight Learning by Perceptron or Delta Rule

$\text{Delta} = \text{Target} - \text{Output}_i$

$\text{weightChange} = \text{LearningRate} \times \text{Delta} \times \text{output}_j(t)$

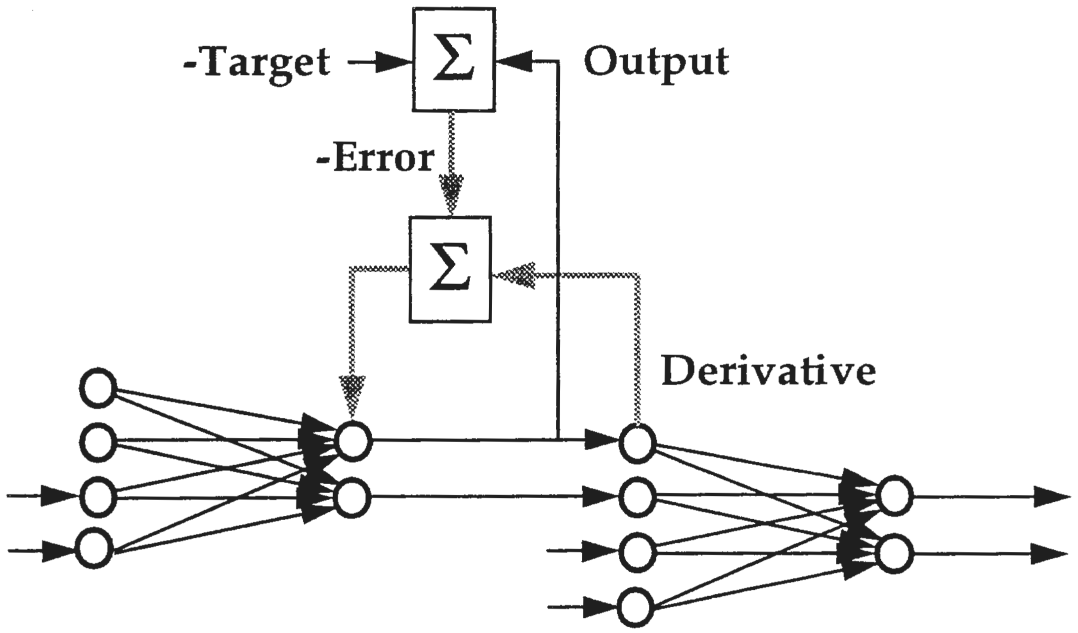
Backpropagation Learning

- For Nonlinear, Multi-layer Perceptron
- Nonlinear Generalization of Delta Rule
- Efficient Computation of Error Gradient in Weight Space for Multiple Layers
- Learns Functional Input/Output Mappings
- Many Enhancements Developed for Faster Learning
    - Momentum
    - Adaptive Learning Rates
    - Conjugate Gradient Learning

Backpropagation Through Time

- Can be used in recurrent networks to propagate error signals "back in time."
- Provides building block for recurrent networks and <u>Approximate Dynamic Programming</u> with neural networks.

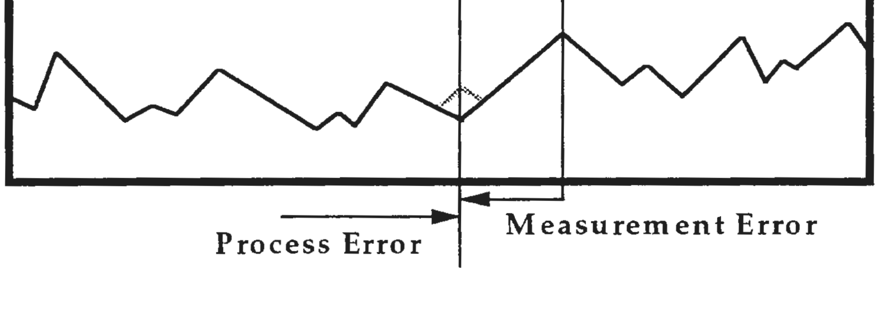
The Pure Robust Technique



The Pure Robust technique is a modified form of time-lagged recurrence where the across-time connection is used to replace the standard time-series input with the previous forecast. In optimal control theory, this is known as parallel system identification. However, in the classical training algorithm, the effect of new learning on past performance in different operating regions is not taken into account. The pure robust method, and a related compromise method, explicitly optimize across all times simultaneously.

The pure robust method functions by separating process error from the measurement error which is determined by the lack of predictiveness in current values. When measurement error occurs, the forecast output is a better unbiased estimate of the true value than the measured value, and is used in its place.

Detecting Measurement Error



Attractor Networks

- Simultaneous Recurrent Network
- Finite Set of Stable States
- Each State is an Attractor for a Set or Range of Inputs
- Autoassociative or Heteroassociative
- Capabilities:
  - Noise Filter
  - Reduce Output Dimensionality (Categorize)
  - Find Optimal States

Boltzmann Machine

- Stochastic Attractor Network
- Nonlinear, Multilayer Generalization of Hopfield Model
- Output transformation includes a 'Temperature' term.
- Output transformation produces the probability that the node is activated.

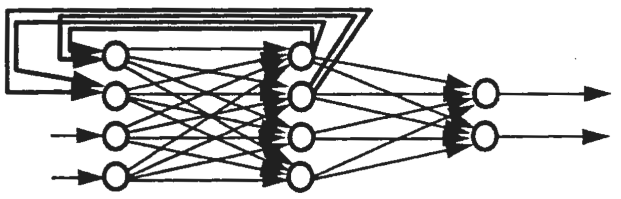
$$\text{Probability} = 1 / (1 + e^{-\text{Input}/\text{Temperature}})$$

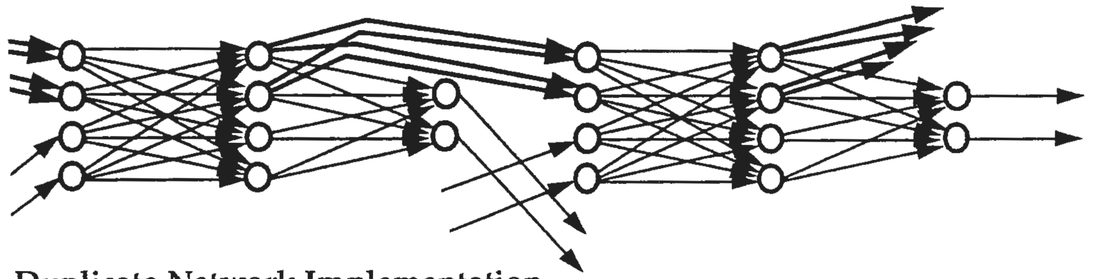
- Exploits *Simulated Annealing* for accuracy and speed by lowering the temperature during convergence

Attractor networks such as the Boltzmann Machine show tremendous potential for the future. However, much work remains to be done in the research lab before this class of architecture is ready for general use. The current formulation requires vast amounts of computing power and provides "disposable," one-use solutions for each unique architecture.

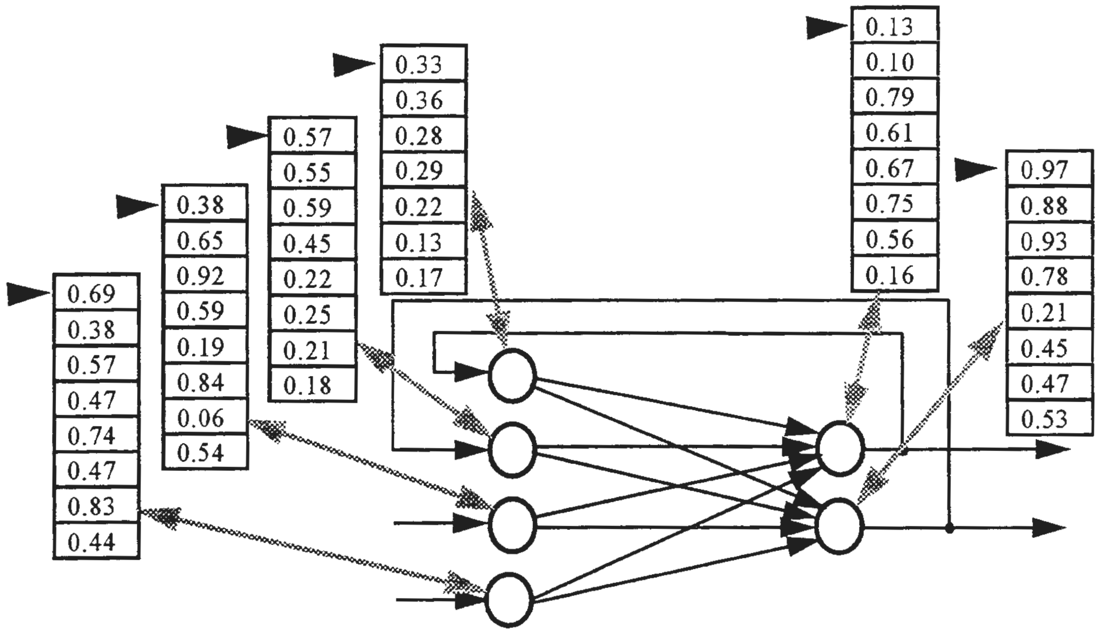
Implementation Options for Time-Lagged Recurrence

Design Model





Duplicate Network Implementation



Activation Stack Implementation

The design model in the first graph can be implemented by duplicating the entire network once for each desired model order. A much more efficient technique only requires storing the previous activation values of each node in a stack, so that they can be passed forward as inputs to future firings and "unwound" during training with the backpropagation-through-time algorithm.

Emulation with Neural Networks

- Emulation: Configuring and adapting a neural network to extract the same information from the data as a classical system

- Requires specifying appropriate inputs and model order

  Does not require specifying mathematical formulation of model.

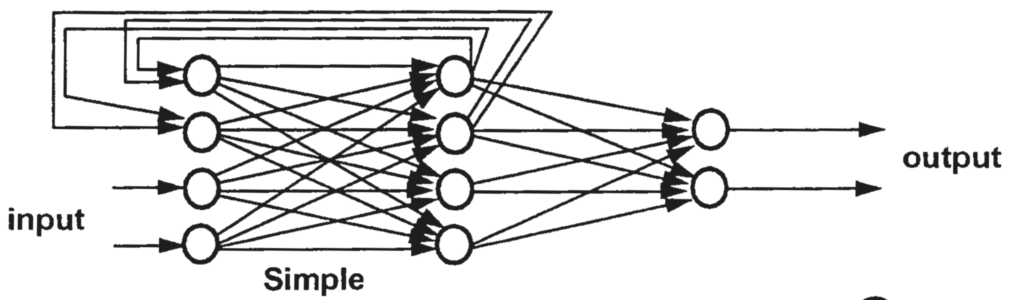
Input/Output Matrix

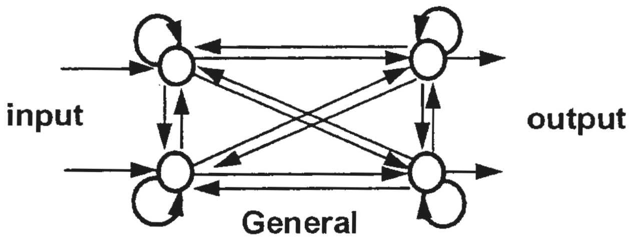
Industry	Energy			
Wkly DJ 20 Bnd Yld	7.35%			
Daily Crude Oil %	0.05			
Day of Year	31			
Day of Week	Monday			
Time of Day	9:00	10:00	11:00	12:00
Price as % of 12	0.8	0.83	0.82	0.77
Actual % Change	0.01	0.01	-0.002	-0.22
Forecast % Change	0.01	0.015	0.023	-0.01
Forecast - Actual	0	0.005	0.025	0.012
Recurrent x	x1	x2	x3	x4
Recurrent y	y1	y2	y3	y4
Recurrent z	z1	z2	z3	z4

Known External Data & Context	Future External Data & Context
Known Network Data	Network Predictions & Context

The input/output matrix exposes the types of data than can be used as input to a neural network to emulate a variety of classical systems. If only data above the heavy line is used as input, a classical time-series or multi-variate regression results. If error data below the line is included as inputs, auto-regression or parallel system identification results. However, if the recurrent connections are included, and a recurrent learning algorithm is used, non-linear Kalman filters can be emulated.

Recurrent Networks



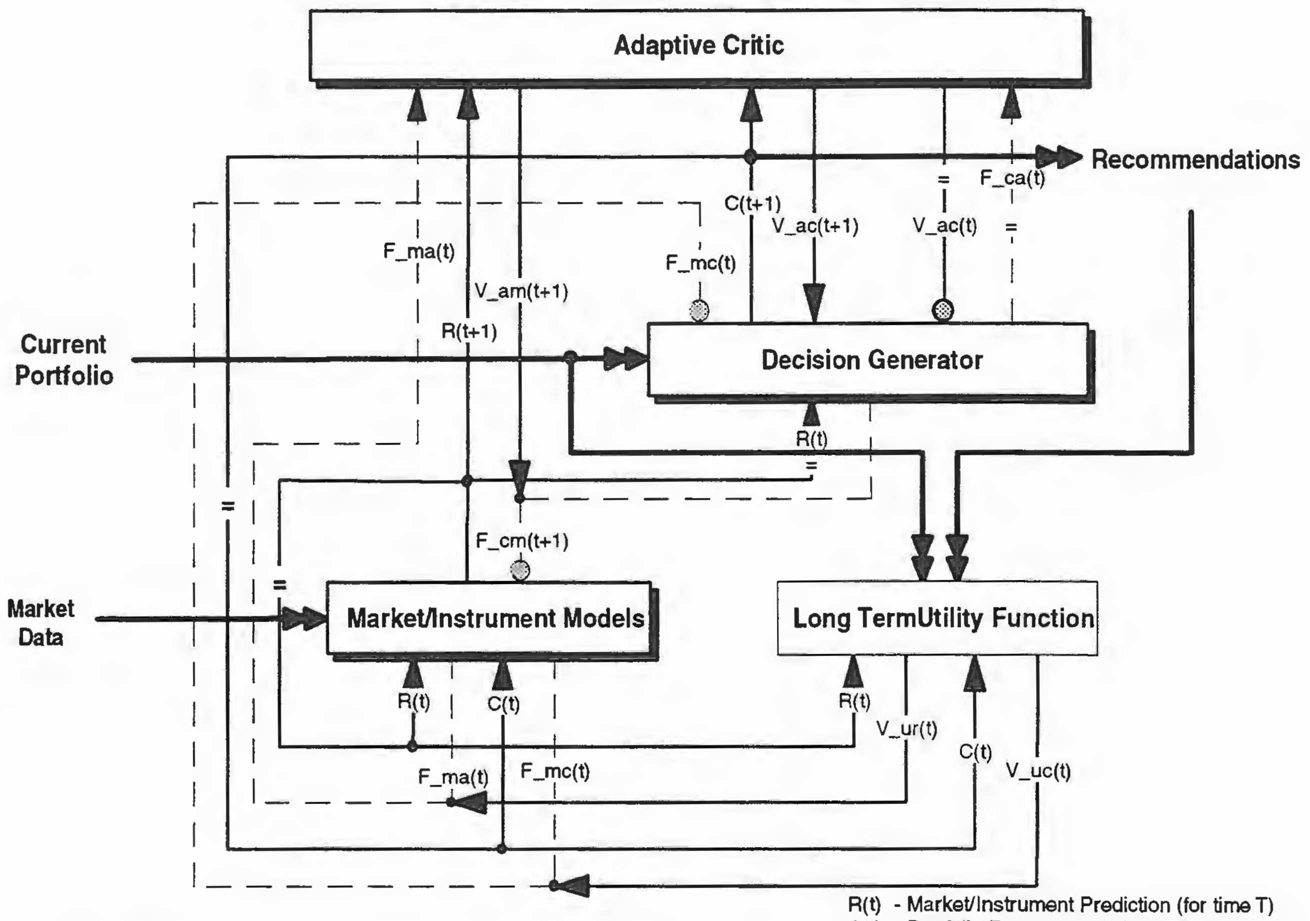


Simultaneous Recurrence
vs
Time-Lagged Recurrence

Recurrent networks contain processing cycles of some form. In time-lagged recurrence, the cycles connect past states of the network to future states in a time series. Depending on the configuration specifics, such a system can emulate high-order non-linear ARMA systems.

Networks with simultaneous recurrence contain processing cycles within each single input/output process. In the most general case, this requires analogue processing elements that dynamically converge to a stable state prior to generating each output.

Multi-Network Architectures



R(t)	- Market/Instrument Prediction (for time T)
C(t)	- Portfolio Recommendation Command Vector
V_xy(t)	- Utility Value from x to y
F_xy(t)	- Feedback from x to y
=	- Time Recurrance

— —▶ Information Flow
——▶ Feedback Correction
— —● Derivative Flowthrough

Adaptive Portfolio Management via Dual Heuristic Programming (patent pending - BHI &Paul Werbos)

This final chart illustrates a high-level design for an approximate dynamic programming solution to portfolio management. This design requires three neural networks, one for modeling the market or markets of interest, one for generating buy-sell decisions, and one for tuning the tactical decisions for long-term profitability. (This architecture has been submitted for patent protection by BHI with Dr. Paul Werbos, the original inventor of the backpropagation algorithm.)

Summary

The presentation covered a wide range of materials covering the early development of neural network technology through the most advanced techniques available. Mastering this material sufficiently to implement advanced financial applications is challenging, to say the least. However, for the adventurous and patient investor, the potential return is enormous.