

Forecasting Financial Markets

Neural Networks

Copyright © 1999-2006
Investment Analytics

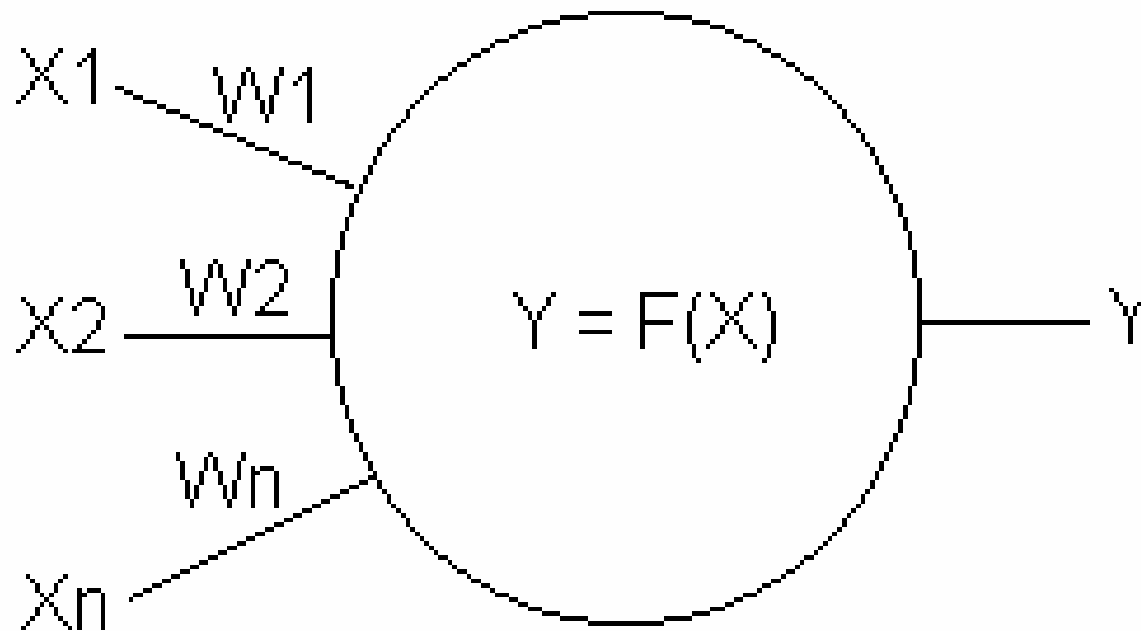
Overview

- Overview of neural networks
- Design considerations
- Applications

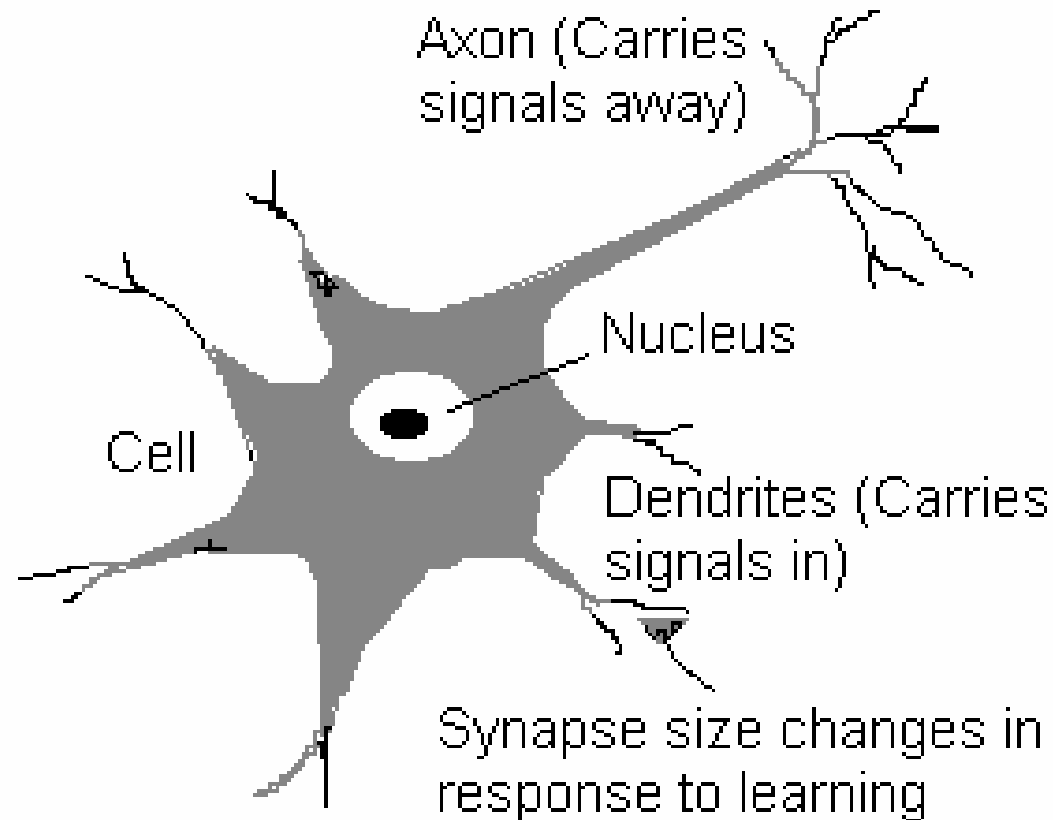
A Neural Network

- Processing elements
 - Neurons
 - Receives & processes input(s)
 - Delivers single output
- Network
 - Collection of interlinked neurons
 - Grouped in layers
 - Input
 - Intermediate (hidden)
 - Output

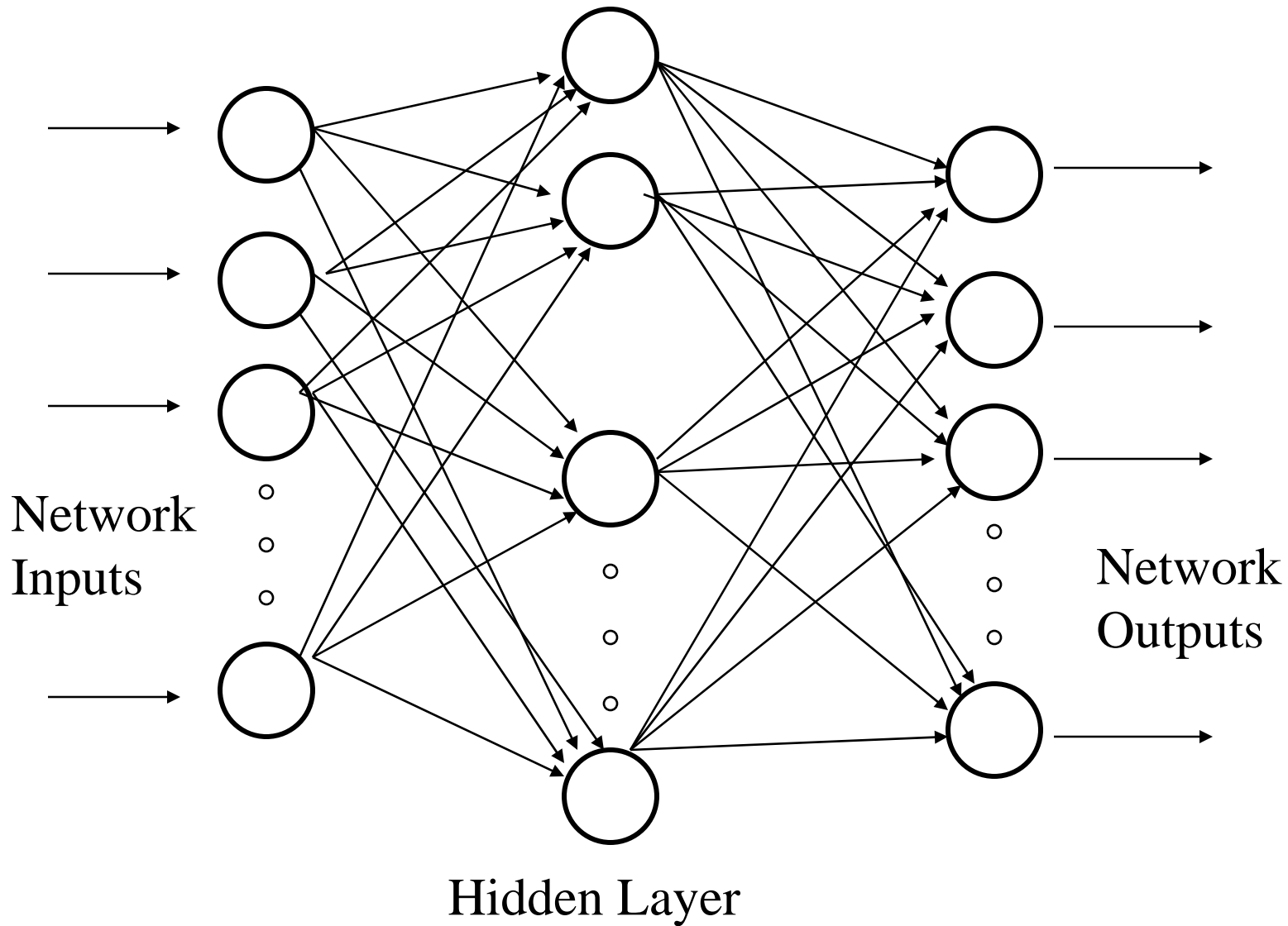
A Schematic Diagram of a Neuron



The Neuron Analogy



A 3 Layer Neural Network



Processing Information

➤ Inputs

- Correspond to single attributes
 - Can include qualitative data

➤ Outputs

- Solution to a problem
 - E.g. forecast, or binary value

➤ Weights

- Express relative importance of data
 - On inputs or data transferred between layers
- “Learning” = adapting weights

Activation Function

- Determines whether a neuron will “fire”
 - I.E. Produce an output
- Weighted sum of inputs
 - For N inputs i into neuron j

$$Y_j = \sum_{i=1}^N W_{ij} X_i$$

Transfer Function

- Transforms or normalizes output
 - Also called a transformation or squashing function
- Popular choice
 - Sigmoid : $f(x) = 1/(1+e^{-x})$
- Alternative: threshold detector / hard limiter
 - E.G. $F(Y_j)$ in range $\{0, 1\}$ if $Y_j > 0.5$, 0 otherwise

Architecture / Network Topology

- Number of neurons
- Number of hidden layers
- Connections
 - Feed forward/backwards
 - Fully or partially connected
- Static or adaptive architecture

Learning

- Supervised
 - Uses set of inputs for which desired output is known
 - Cost function $f(\text{desired-actual})$ used to change weights
 - Example: Hopfield network
- Unsupervised
 - Network shown only inputs
 - No information on “correct” outputs
 - Self-organizing
 - Example: Kohonen self-organizing feature maps

Training

- Data divided into training & testing data sets
 - Training set used to adapt weights
 - Many iterations or “epochs”
 - Training time dependent on data, network architecture, learning algorithm
 - Forecasting performance tested on test data set
 - Cost function comparing desired vs. Actual outputs
 - Stopping rule
 - Determines when to terminate training
 - When weights stabilize
 - When cost function minimized
 - Danger of over-fitting

Applications in Finance

- Bankruptcy prediction
- Bond rating
- Consumer credit scoring
- Financial market forecasting
 - Equities, currencies, commodities, bonds, derivatives
 - Security selection
 - Portfolio optimization
 - Trading systems

A Simple NN Example

- Supervised Learning of OR operator
 - Inputs X_1, X_2
 - Outputs Z (desired), Y (actual)
 - Weights W_1, W_2 ; initial values 0.1 and 0.3
 - Transfer Function
 - $F(Y) = 1$ if $Y > \text{Threshold Value (0.5)}$; 0 otherwise
 - Learning
 - $\Delta = (Z - Y)$
 - $W_i(\text{final}) = W_i(\text{initial}) + \alpha\Delta X_i$
 - α is learning coefficient (0.2)

A Simple NN Example

| Iteration 1 | | Initial | | | | | Final | | | |
|-------------|-------|---------|-------|-------|-----|----------|-------|-------|--|--|
| X_1 | X_2 | Z | W_1 | W_2 | Y | Δ | W_1 | W_2 | | |
| 0 | 0 | 0 | 0.10 | 0.30 | 0 | 0 | 0.10 | 0.30 | | |
| 0 | 1 | 1 | 0.10 | 0.30 | 0 | 1 | 0.10 | 0.50 | | |
| 1 | 0 | 1 | 0.10 | 0.50 | 0 | 1 | 0.30 | 0.50 | | |
| 1 | 1 | 1 | 0.30 | 0.50 | 1 | 0 | 0.30 | 0.50 | | |

| Iteration 4 | | Initial | | | | | Final | | | |
|-------------|-------|---------|-------|-------|-----|----------|-------|-------|--|--|
| X_1 | X_2 | Z | W_1 | W_2 | Y | Δ | W_1 | W_2 | | |
| 0 | 0 | 0 | 0.70 | 0.70 | 0 | 0 | 0.70 | 0.70 | | |
| 0 | 1 | 1 | 0.70 | 0.70 | 1 | 0 | 0.70 | 0.70 | | |
| 1 | 0 | 1 | 0.70 | 0.70 | 1 | 0 | 0.70 | 0.70 | | |
| 1 | 1 | 1 | 0.70 | 0.70 | 1 | 0 | 0.70 | 0.70 | | |

Design Considerations

- Network performance
- Control mechanisms
 - Choice of activation function
 - Choice of cost function
 - Network architecture
 - Gradient descent/ascent efficiency
 - Learning times

Network Performance Measures

- Convergence
 - Accuracy of model fitness in-sample
- Generalization
 - Accuracy of model fitness out-of-sample
- Stability
 - Variance in prediction accuracy

Convergence

- Is network capable of learning classification?
 - Under what conditions?
 - What are computation requirements?
- Fixed topology networks
 - Prove convergence by showing error tends to zero in limit as $t \rightarrow \infty$
 - Using gradient descent
- Other networks
 - Show that network can classify the maximum # possible mappings with arbitrarily large probability

Generalization

- Ability to classify data outside training set
 - Most important performance criterion
- Analogy with curve fitting
 - Two problems
 - Finding order of polynomial
 - Estimating coefficients
 - Too low order (NN structure too simple)
 - Bad approximation both in- and out-of-sample
 - Too high order (NN structure too complex)
 - “Over-fitting” : fits test data well, but out-of-sample performance poor

Stability

- Consistency of results
 - When network parameters are varied
 - Networks often vary widely in predictive performance
 - “Chaotic”: highly sensitive to initial conditions
- Two components of error
 - Bias: due to parameterization & associated assumptions
 - Variance: sensitivity to changes in estimated parameters
 - Regression: high bias, low variance
 - Neural networks: low bias, high variance
 - No parameterization, but may fit entire family of polynomials to given data set

Choice of Activation Function

➤ Sigmoid Functions

- Differentiable and well behaved

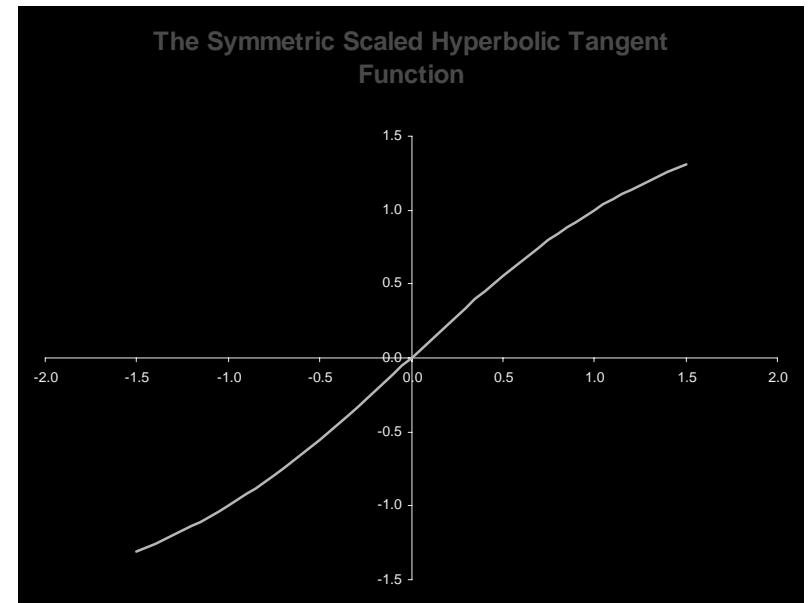
➤ Symmetric

- Typical: scaled hyperbolic tangent

$$f(y) = A \tanh(Sy) = A \frac{e^{Sy} - e^{-Sy}}{e^{Sy} + e^{-Sy}}$$

$$= A - \frac{2A}{1 + e^{2Sy}}$$

- A is amplitude
- S is slope at origin



Choice of Activation Function

➤ Choice of sigmoid parameters

- $A = 1.7159, s = 2/3$

- $F(-1) = -1$ and $f(1) = 1$

- Gain in squashing transformation is normally around 1

- $\Delta^2 f / \delta x^2$ is max at ± 1

- Improves convergence at end of learning session

➤ Symmetric vs. Asymmetric sigmoid functions

- Refenes & Alippi (1991)

- Symmetric functions capable of speed of convergence over asymmetric functions by factor of 10

Cost Function

➤ Quadratic cost function is most common

- Least mean square error

$$E = \frac{1}{2} \sum_{i=1}^n (d_i - y_i)^2$$

- Y_i is current output from unit i
- d_i is desired output from unit i

- Discounted least square error

$$E = \frac{1}{n} \sum_{i=1}^n \frac{1}{1 + e^{(a+bi)}} (d_i - y_i)^2$$

Learning

- Gradient descent used to minimize cost function
 - Change weights in proportion to $\delta_i = \delta e / \delta W_i$
 - $\Delta W_{ij}(t+1) = \lambda \delta_i y_{ij}$
- Learning rate (step size, momentum) λ
 - As $\lambda \rightarrow 0$ and $t \rightarrow \infty$ this procedure will find MSE_{Min}
- Difficult to find appropriate rate
 - Too small
 - Slow convergence
 - May get trapped in local minima
 - Too large: unstable weights

Learning Rate

- Optimal learning rate pattern
 - Smooth MSE chart (for each layer)
 - Smooth weight histograms (for each layer)
- Learning rate adjustment
 - One rate for entire network
 - Different rates for each layer
 - Different rate for each weight

Learning Rate Rules of Thumb

- If no connections that jump layers
 - Learning rate for hidden layer $\lambda_L = 0.5 \lambda_{L+1}$
- With connections that jump layers
 - Learning rate for hidden layer $\lambda_L = 0.75 \lambda_{L+1}$
- Check sign of consecutive weight changes
 - If same, increase λ
 - If opposite, decrease λ
- If MSE chart is erratic
 - Reduce learning rate (for that layer)

Network Architecture

➤ Hidden units

- In general, the fewer the better
 - Network will generalize better
- Another approach: weight sharing
 - Imposing equality constraints amongst connections strengths
 - Reduces # free parameters while preserving network size and ability to recognize complex patterns

➤ Hidden layers

- Typically start with one
- If use more than one make sure to connect each layer to all prior layers

Network Architecture

- Constructive techniques
 - Hidden units added incrementally
- Pruning techniques
 - Attempts to eliminate redundant units
- Genetic algorithms
 - Selects “fittest” of several competing networks

Constructive Techniques

➤ Tiling algorithm

- Divide training data set into “faithful” & “unfaithful” classes
 - I.E. Those the network recognizes correctly and those it doesn't
- Add ancillary unit and connect to layer above
- Select one unfaithful class and train new unit to subdivide it into faithful and unfaithful classes
- Repeat until no unfaithful classes remain
 - Always possible - worst case: one unit for each input pattern
- Add new master output unit and connect to all layers
 - Training the new unit to learning mapping to desired output

Other Constructive Techniques

- Cascade algorithm
 - Adds hidden unit to maximize magnitude of correlation between new unit's output and residual error signal to be minimized
- Dynamic node creation
 - Add new unit is rate of error decrease falls below certain value

Pruning Techniques

➤ Multi-stage stage pruning

- Outputs of hidden units analysed to see if any are not contributing to solution
 - Output of unit doesn't change for any input pattern
 - If output from two units is identical or opposite (for all inputs)
- Repeat for next layer

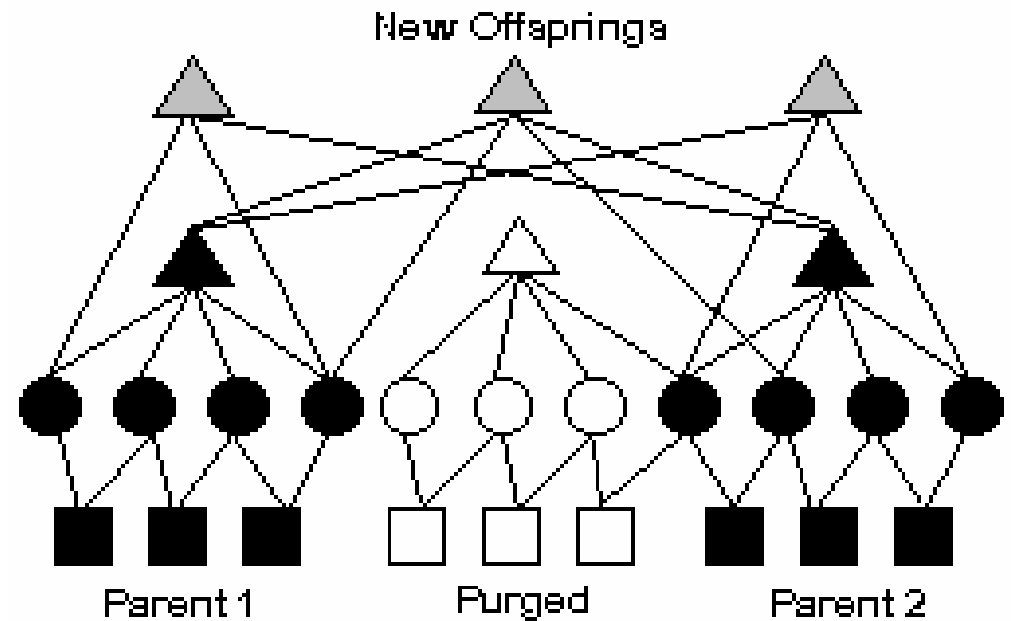
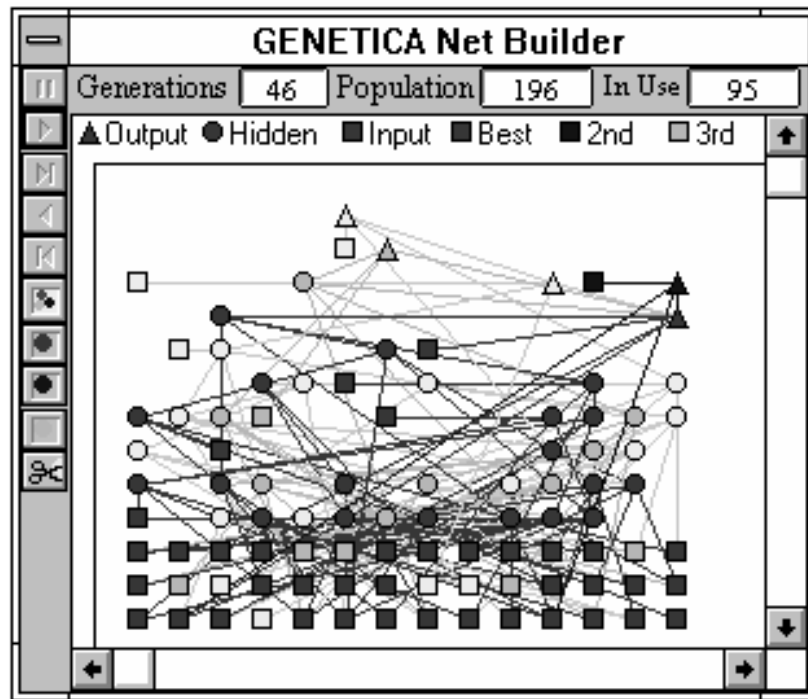
➤ Weight decay

- Weights without much influence subjected to time decay
- Equivalent: add penalty term to cost function
 - $E^* = \text{MSE} + b\sigma\sigma w_{ij}^2$

Genetically Evolved Neural Networks

- Initial population of randomly generated networks
- Proceed through training cycle with all networks
- At end of initial training cycle
 - Worst performing networks are deleted
 - Best performing networks are “mated”
- Continue training with all networks
- Occasional random mutations introduced
 - Randomize weights of lowly ranked networks
 - Change forecast horizon, lags on input variables

Genetic Evolution of a Neural Network



Training

- Epoch
 - # Training cycles after which weights are updated
- Determining epoch size
 - Start with initial epoch
 - Train network for large # (10,000) iterations
 - Test network and record R^2 (for each output)
 - Repeat for variety of epoch sizes
 - Pick epoch size that maximizes R^2
- Controlling over-fitting
 - Terminate training when MSE on *test* set starts to rise

Initial Weights

- Start with unequal initial weights
 - Rumelhart, Hinton, Williams (1986)
 - Will not converge if solution requires unequal weights
- Testing stability
 - Initial weight matrix defines starting point on the weight-error surface
 - Need several training sets with different random weights to test for statistical stability

Data Modeling

➤ Detrending

- Removing of seasonality and trends to achieve stationarity

➤ Normalization

- Variables scaled to have zero mean, unit SD
 - Brings inputs into normal operating range of activation function
 - Otherwise activation values may tend to zero
 - Network paralysis

$$X_i'(t) = \frac{X_i(t) - \bar{X}_i}{\sigma_{X_i}}$$

Data Modeling

➤ Scaling of Outputs

- Some transfer functions reach max/min values only when inputs reach infinity

$$Y'(t) = SCALE \times Y(t) + OFFSET$$

$$SCALE = \frac{MAX - MIN}{Y_{Max} - Y_{Min}}$$

$$OFFSET = MAX - \frac{MAX - MIN}{Y_{Max} - Y_{Min}} Y_{Max}$$

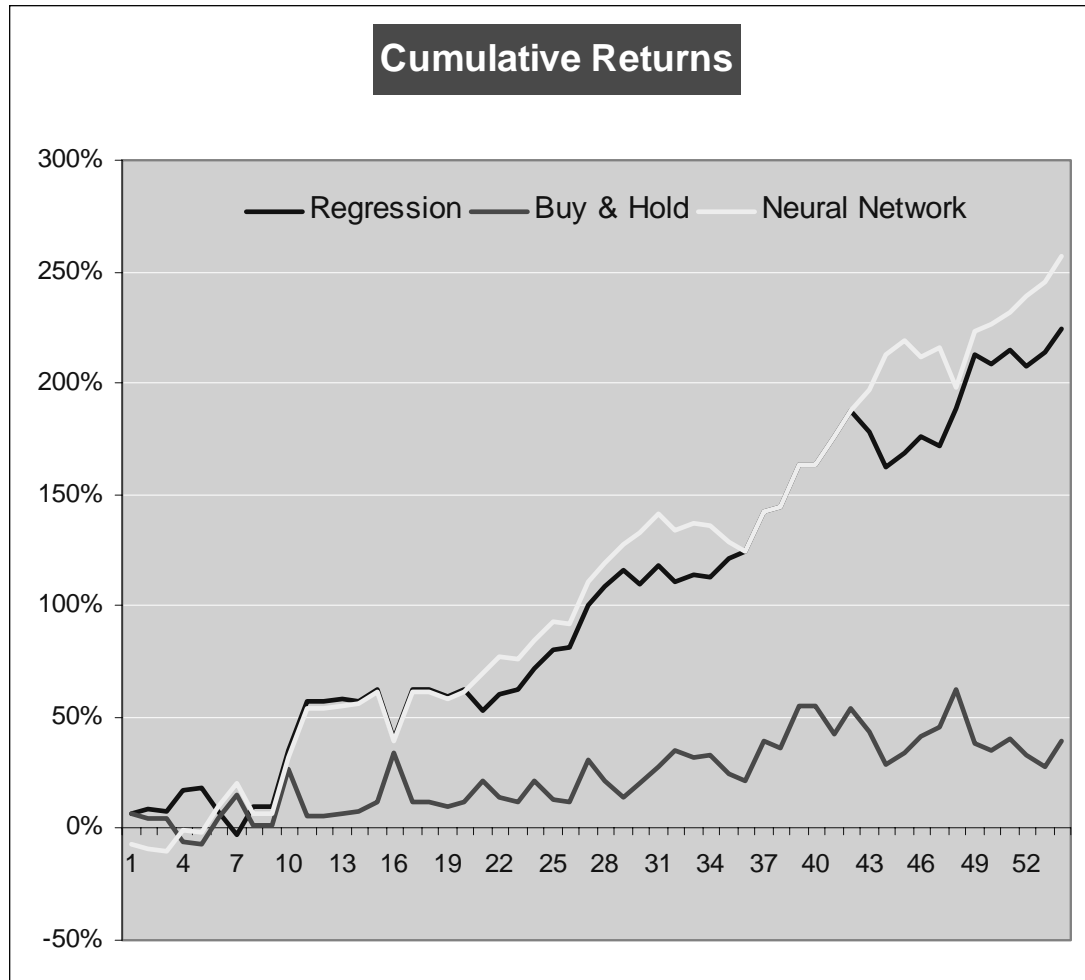
➤ Multi-Collinearity

- Independent variables are correlated
 - Solution: use principal components analysis to orthogonize inputs

Lab: Modeling Implied Volatility on IBEX Options

- Compare two forecasting techniques
 - Regression
 - Genetic neural network
- Forecast implied volatility
 - Evaluate trading performance

Solution: Modeling Volatility on IBEX Options



Solution: Modeling Volatility on IBEX Options

| Out of Sample - Regression Model | | | | Out of Sample - Neural Network | | | |
|----------------------------------|---------------|-----------------------------------|---------|--------------------------------|---------------|-----------------------------------|---------|
| AIC | 483.05 | Total Return | 224.71% | AIC | 481.12 | Total Return | 256.50% |
| BIC | 506.99 | Excess over Buy & Hold | 185.70% | BIC | 505.06 | Excess over Buy & Hold | 217.49% |
| R² | 42.9% | Average Winner/Loser | 1.57 | R² | 29.7% | Average Winner/Loser | 1.57 |
| Adj. R² | 35.8% | Directional Change | 65.5% | Adj. R² | 21.0% | Directional Change | 65.5% |
| MSE | 135.6 | Correlation Coefficient | 63.2% | MSE | 130.9 | Correlation Coefficient | 53.5% |
| MAD | 8.83 | Mean Reversion | 0.89 | MAD | 8.10 | Mean Reversion | 0.87 |
| MAPE | 169.7% | Distance from Ideal | 51.6% | MAPE | 135.7% | Distance from Ideal | 57.9% |
| Theils U | 0.83 | Sharpe ratio | 0.42 | Theils U | 0.79 | Sharpe ratio | 0.49 |
| DW | 1.41 | Luck Coefficient (5%) | 27.9% | DW | 2.34 | Luck Coefficient (5%) | 27.9% |

Summary: Neural Networks

➤ Pros

- Can capture non-linear effects
 - Pattern recognition
- Process model not required
- Wide range of applications in finance

➤ Cons

- ‘Black-box’ approach
- Sometimes poor stability & generalization characteristics