Artificial Intelligence

25th April 2017

Varun Natu
Riyank Varma
Kartik Jindal

Stock Index Forecasting using ANNs

Synopsis

Main Goal: Prediction of NSE NIFTY 50 Index price movement

Artificial Intelligence Concepts to be Implemented

- Artificial Neural Networks

Data Sources

- Yahoo Finance : Jan 1st 2010 - December 31st 2016

Preview

- Data
    - Features : What? Why?
    - Preprocessing : How ?
    - Relation to objective
- Neural Network Sample Space
    - Network Structures
    - Inner Details : Activation and Optimisation
- Results
    - What did well? How well?
    - The Network we Chose
- Practical Results
    - Can we generate returns? How much returns?
- Further Work
    - Extending the work.

Tools

- MATLAB Neural Network Toolbox
- Data from Yahoo Finance
- Literature Review of Neural Nets in Market Prediction
- Preprocessing using Excel and MATLAB

Objectives and Uses
Technical Perspective

- To study the dependence of the network on
  - Activation Functions
  - Optimisation Techniques
- We have trained and tested **240** different Networks using Backpropogration
  - 3 Activation functions
  - 5 Optimisation Techniques
  - 16 Network Architectures (having 1 to 5 hidden layers)
- As an error metric : Mean Squared Error (MSE)
- As a network performance scale : Determination Coefficient (R-Squared)

Network Architectures

No.	Structure
1	2
2	5
3	5-5
4	5-10
5	10-10
6	10-20
7	40-40
8	50-100
9	100-200
10	200-300
11	20-40-20
12	20-50-20
13	50-100-50
14	20-40-40-20
15	10-20-20-10
16	10-20-20-20-10

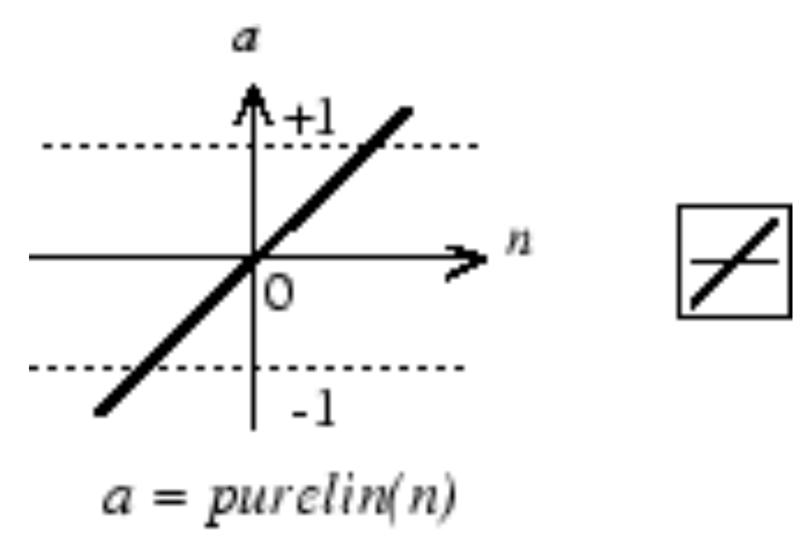
- Tried to accommodate diverse networks
  - Heavy Layers
  - Light Layers
  - Could not try some of the more computationally expensive networks due to explosion of training time
    - 200-300 Structure took 30-35 minutes for just a few iterations

Activation Functions

- The function used to calculate the output of a node within a layer
- We have kept the activation function the same across all layers of a particular network
    - Intra-Network Variation of Activation Functions
- Our literature review led us to 3 prominent functions
    - Linear Output (Identity) ***purelin()***
    - Logarithmic Sigmoid (Regular Sigmoid) ***logsig()***
    - Tan Sigmoid (Arctan) ***tansig()***

Linear Output

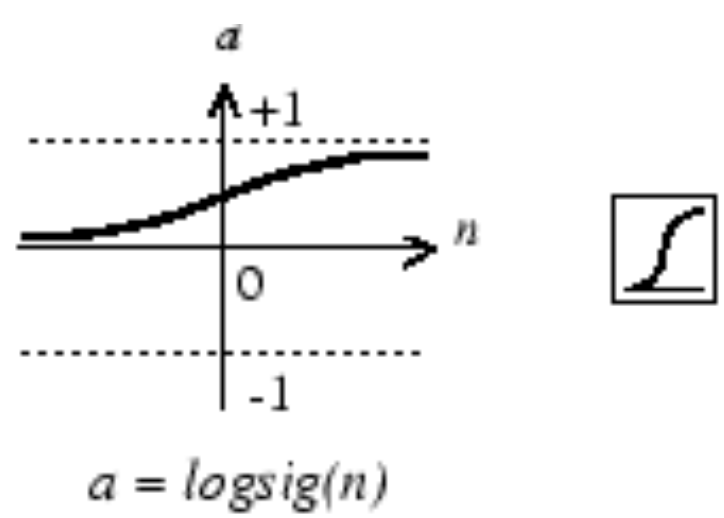
- Identity or Linear Activation Function
  - Outputs a linear combination of the inputs
  - Used primarily for function firing problems
  - Fast and Simplest to use



Linear Transfer Function

Logarithmic Sigmoid

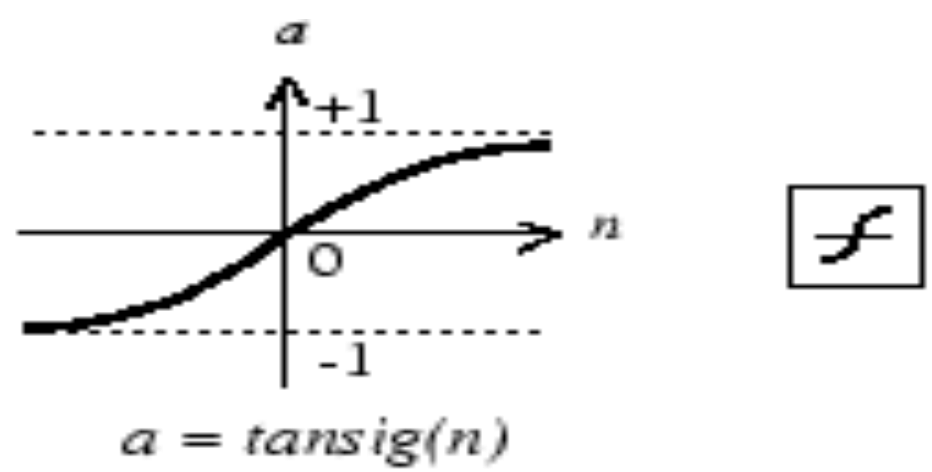
- One of the most popular non-linear activation function for NNs
  - Generates a value between 0 and 1
  - It is found to be useful for positive target values
  - Known to be beneficial for pattern matching Problems



Log-Sigmoid Transfer Function

Tangent Log Sigmoid

- It is the arctan of ‘tanh’ function
  - Generates a value between -1 and 1
  - Used for models with real values
  - It has been found to be useful for models which tend to be quite non-linear



Tan-Sigmoid Transfer Function

Optimisation Techniques

- We wanted to see the effect of optimisation techniques on network performance so took 5 methods other than Simple Gradient Descent
    - Gradient Descent with Momentum
    - Gradient Descent with Adaptive Learning Rate
    - Levenberg-Marquardt Optimisation
    - One Step Secant Method
    - Scaled Conjugate Gradient

Gradient Descent with Momentum

- Used quite widely for deeper networks and higher number of nodes per layer
- Momentum allows a network to respond not only to the local gradient, but also to recent trends in the error surface
  - Avoid shallow local minimums
  - Done by adding a fraction of the previous change to weights
- Results in faster Convergence and Lesser Oscillation

Gradient Descent with Adaptive Learning Rate

- In standard gradient descent, the performance of the algorithm is very sensitive to the proper setting of the learning rate
    - Too high : Oscillation and Instability
    - Too small : Very slow convergence
- An adaptive learning rate will attempt to keep the learning step size as large as possible while keeping learning stable.
    - New Error > Old Error by more than a predefined ratio : new weights discarded, learning rate lowered
    - New Error < Old Error : Weights kept, learning rate increased

Scaled Conjugate Method

- A search is performed along conjugate directions, which produces generally faster convergence than steepest descent direction

- May require more iterations to converge than other CG algorithms

    - Computation per iteration is much lesser due to avoidance of line search

    - Uses step size scaling mechanism rather than line search per learning iteration.

Levenberg-Marquardt Method

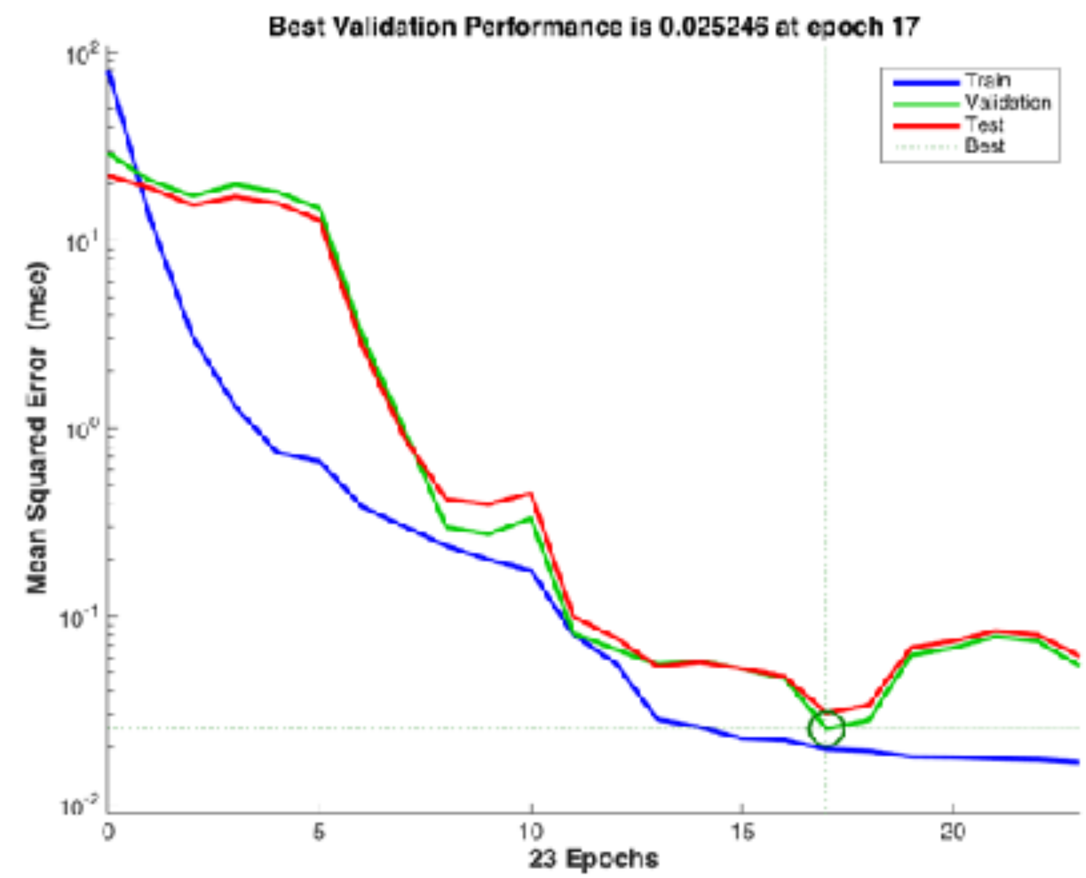
- Often the fastest algorithm
  - More memory intensive than other choices
- Designed to approach second-order training speed without having to compute the Hessian matrix
  - Hessian is estimated with the Jacobian J if the performance function is MSE (this is what we have used)

One Step Secant Method

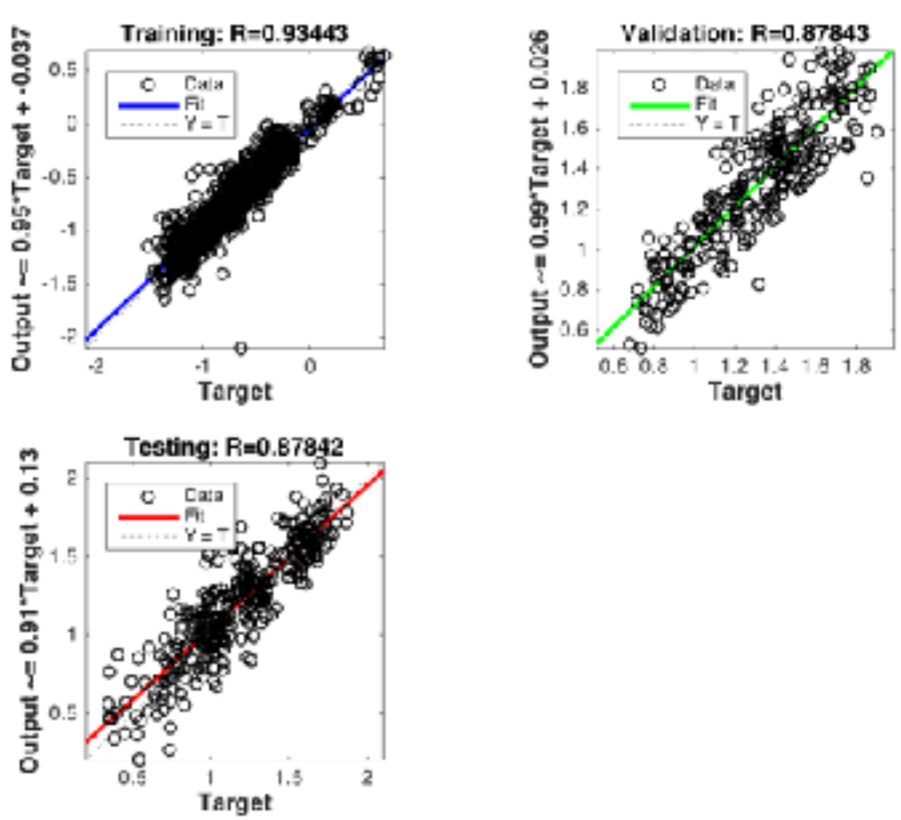
- An attempt to bridge the gap between Conjugate Gradient Algorithms and Newton’s Method
  - Avoids memory and time associated with Hessian storage of Newton
  - It assumes that at each iteration, the previous Hessian was the identity matrix
    - Allows for computation of new direction without an inverse operation on the Hessian

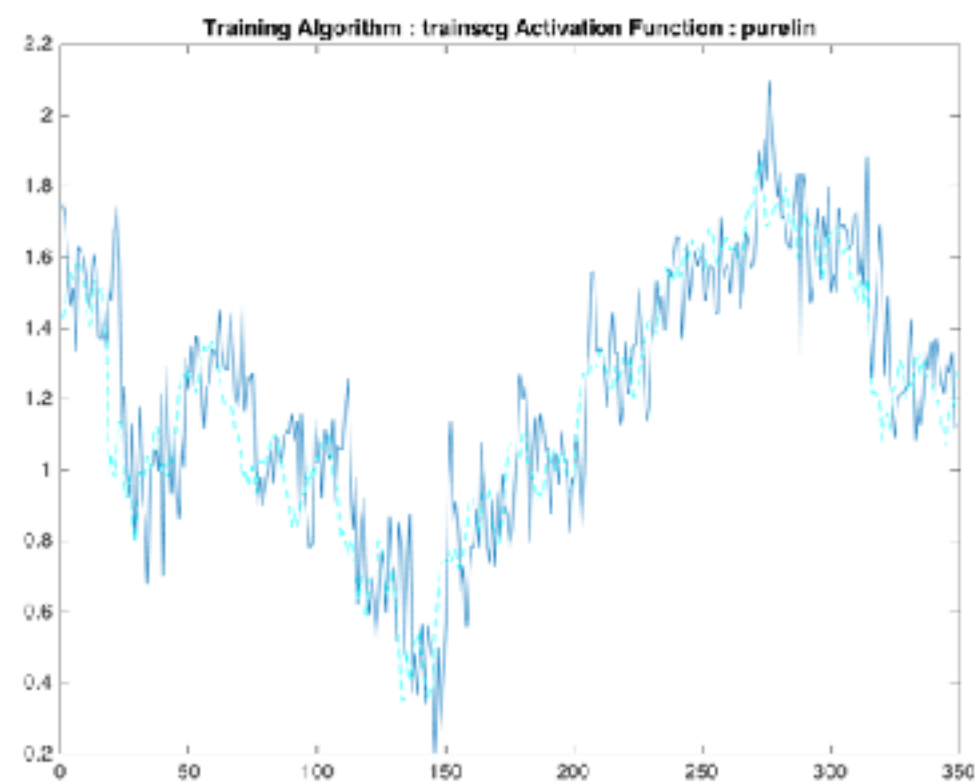
Results

- For each architecture we've plotted 3 sets of graphs
    - Regression Graphs
    - Error Graph with iterations
    - Test Data Performance



Example : For 2 Layer 10-20 Network Using SCG and Linear Activation





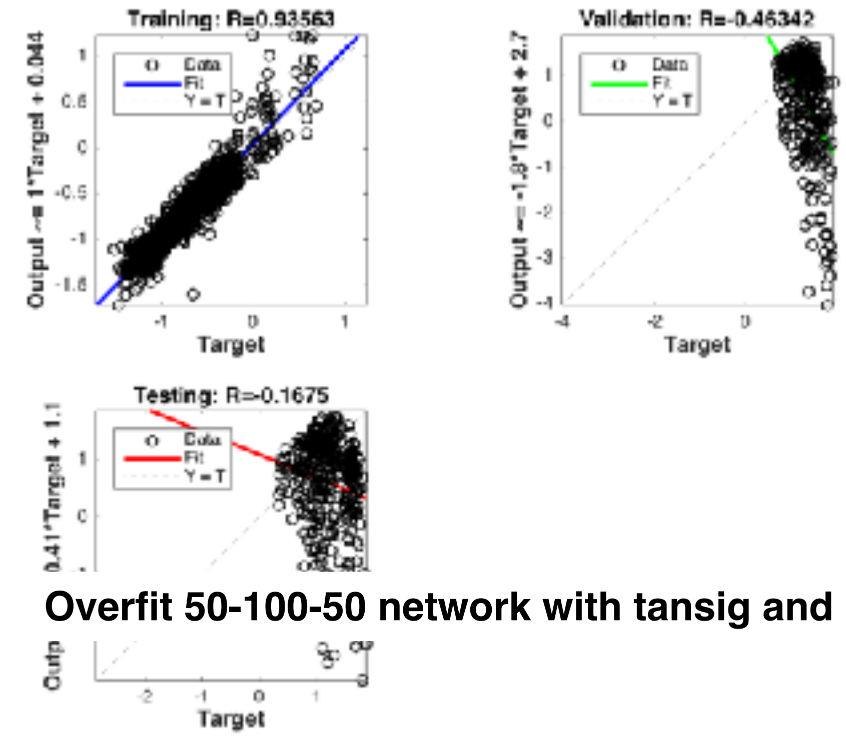
Results : Within Architecture

- Levenberg- Marquardt repeatedly outperformed the other optimisation techniques
    - Promising results with simpler and complicated structures both
- Linear Activation suited simpler architectures (fewer layers/ nodes) much better
- SCG also was seen as giving comparable results but only for much simpler networks

Architecture	Training Algorithm	Activation Function	R-Squared on Test
5-5	Adaptive Weights	Pure Linear	0.69
5-5	Adaptive Weights	Tan Sigmoid	0.227
5-5	Momentum	Pure Linear	-0.245
5-5	Momentum	Tan Sigmoid	-0.174
5-5	Levenberg-Marquardt	Pure Linear	0.983
5-5	Levenberg-Marquardt	Tan Sigmoid	0.94
5-5	One Step Secant	Pure Linear	0.760
5-5	One Step Secant	Tan Sigmoid	0.687
5-5	Scaled Conjugate	Pure Linear	0.979
5-5	Scaled Conjugate	Tan Sigmoid	0.482

Architecture	Training Algorithm	Activation Function	R-Squared on Test
50-100-50	Adaptive Weights	Logarithmic Sigmoid	0.626
50-100-50	Adaptive Weights	Pure Linear	0.432
50-100-50	Adaptive Weights	Tan Sigmoid	0.814
50-100-50	Momentum	Logarithmic Sigmoid	-0.491
50-100-50	Momentum	Pure Linear	-0.085
50-100-50	Momentum	Tan Sigmoid	0.702
50-100-50	Levenberg-Marquardt	Logarithmic Sigmoid	0.756
50-100-50	Levenberg-Marquardt	Pure Linear	0.973
50-100-50	Levenberg-Marquardt	Tan Sigmoid	0.058
50-100-50	One Step Secant	Logarithmic Sigmoid	0.112
50-100-50	One Step Secant	Pure Linear	0.028
50-100-50	One Step Secant	Tan Sigmoid	-0.173
50-100-50	Scaled Conjugate	Logarithmic Sigmoid	-0.033
50-100-50	Scaled Conjugate	Pure Linear	0.368
50-100-50	Scaled Conjugate	Tan Sigmoid	-0.167

Observations



Overfit 50-100-50 network with tansig and scg

- Simpler Networks tend to the job just as well
  - We repeatedly found that the determination coefficient values that were obtained on the test data were just as good if not better for simpler network of 1-2 layers
  - Added Training time not worth the gains
    - Time may be spent instead in collecting inputs (discussed in Further Work - sentiment analysis?)
- Contrary to what we expected : Linear activation performed the best overall
  - Non-Linear features didn't add any power
  - Financial Data reflects time series
  - Trends observed are simple curves not requiring higher degrees of non-linearity
  - logsig and tansig often led to overfitting as did complicated networks

Surprise: 5-5

A two-hidden layer network with 5 nodes each was chosen by us as the best network over it's determination coefficient on the testing network

Optimisation : Levenberg-Marquardt

Activation Function : Linear Output