Stock Prediction using Artificial Neural Networks

Abhishek Kar (Y8021),
Dept. of Computer Science and Engineering, IIT Kanpur

Abstract—**In this work we present an Artificial Neural Network approach to predict stock market indices. We outline the design of the Neural Network model with its salient features and customizable parameters. A number of activation functions are implemented along with options for crossvalidation sets. We finally test our algorithm on the Nifty stock index dataset where we predict the values on the basis of values from the past *n* days. We achieve a best case accuracy of 96% on the dataset.**

Index Terms--**Artificial neural networks, Image sequence analysis, Multi-layer neural network, Prediction methods, Stock markets.**

I. INTRODUCTION

Artificial Neural Networks have seen an explosion of interest over the last few years, and are being successfully applied across an extraordinary range of problem domains, in areas as diverse as finance, medicine, engineering, geology and physics. There have been many attempts to formally define neural networks.

"A neural network is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes" – DARPA Neural Network Study (1988)

"A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1. Knowledge is acquired by the network through a learning process.
2. Interneuron connection strengths known as synaptic weights are used to store the knowledge." - Haykin (1994)

"A neural network is a circuit composed of a very large number of simple processing elements that are neurally based. Each element operates only on local information. Furthermore each element operates asynchronously; thus there is no overall system clock." – Nigrin (1993)

"Artificial neural systems, or neural networks, are physical cellular systems which can acquire, store, and utilize experiential knowledge." – Zurada (1992)

A three layer neural network has been proved to be a universal function approximator and finds its use in a number of fields like sales forecasting, data validation, customer research, price forecasting, medicine etc.

Neural networks grew out of research in Artificial Intelligence; specifically attempts to mimic the fault-tolerance and learning capacity of biological neural systems by modeling the low-level structure of the brain. They signify the connectionist approach in AI where phenomena are emergent processes of interconnected networks of simple units. Upon failure of expert systems (based on the symbolic approach), it was apparent that it would be necessary to build systems that mimic the architecture of the human brain.

The rest of the paper is organized as follows. Section II provides a description of Artificial Neural Networks. In Section III we describe the backpropagation algorithm for training ANNs. In Section IV we describe our trained ANN in detail. In Section V we describe the stock dataset, and the preprocessing used. In Section VI we provide the results of simulations on the above datasets, along with a few others. We conclude the paper in Section VII.

II. ARTIFICIAL NEURAL NETWORKS

In this section we describe the structure of Artificial Neurons and how they are connected to construct Artificial Neural Network.

A. Artificial Neurons

Artificial neurons are inspired from biological neuronal structure. The transmission of a signal from one neuron to another through synapses is a complex chemical process in which specific transmitter substances are released from the sending side of the junction. The effect is to raise or lower the electrical potential inside the body of the receiving cell. If this graded potential reaches a threshold, the neuron fires. It is this characteristic that the artificial neuron model attempt to reproduce. The neuron model shown in Figure 1 is the one that widely used in artificial neural networks with some minor modifications on it.
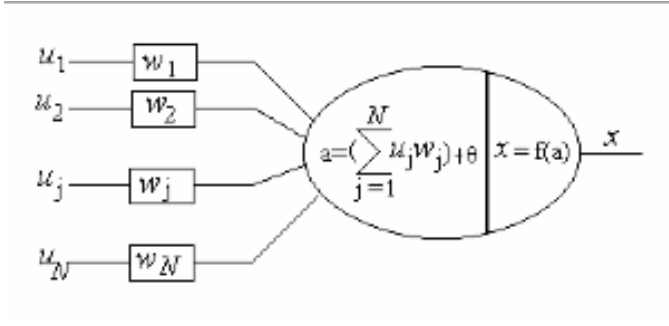
Figure 1: Artificial Neuron Structure

The artificial neuron given in this figure has N input, denoted as $u_1, u_2, ...u_N$. Each line connecting these inputs to the neuron is assigned a **weight**, which are denoted as $w_1, w_2,..., w_N$ respectively. Weights in the artificial model correspond to the synaptic connections in biological neurons. The **threshold** in artificial neuron is usually represented by θ and the **activation** corresponding to the graded potential is given by the formula:

$$a = \sum_{j=1}^{N} w_j u_j + \theta$$

The inputs and the weights are real values. A negative value for a weight indicates an **inhibitory** connection while a positive value indicates an **excitatory** one. Although in biological neurons, has a negative value, it may be assigned a positive value in artificial neuron models. Sometimes, the threshold is combined for simplicity into the summation part by assuming an imaginary input $u_0 = +1$ and a connection weight $w_0 = \theta$. Hence the activation formula becomes:

$$a = \sum_{j=0}^{N} w_j u_j$$

The output value of the neuron is a function of its activation in an analogy to the firing frequency of the biological neurons:

$$x = f(a)$$

There are a number of functions that are used. Some include binary threshold, linear threshold, sigmoid, hyperbolic tan and Gaussian.
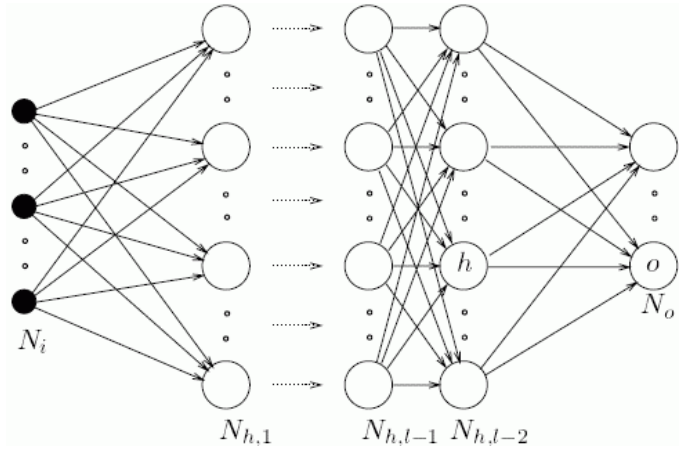
B. Artificial Neural Networks

While a single artificial neuron is not able to implement some boolean functions, the problem is overcome by connecting the outputs of some neurons as input to the others, so constituting a neural network. Suppose that we have connected many artificial neurons that we introduced in Section 1.2 to form a network. In such a case, there are several neurons in the system, so we assign indices to the neurons to discriminate between them. Then to express the activation i^{th} neuron, the formulas are modified as follows:

$$a_i = \sum_{j=1}^{N} w_{ji} x_j + \theta_i$$

where x_j maybe the output of another neuron or an external input.

There are a number of architectures in use for ANNs. In **feedforward** neural networks, the neurons are organized in the form of layers. The neurons in a layer get input from the previous layer and feed their output to the next layer. In this kind of networks connections to the neurons in the same or previous layers are not permitted. The last layer of neurons is called the output layer and the layers between the input and output layers are called the hidden layers. The input layer is made up of special input neurons, transmitting only the applied external input to their outputs. In a network if there is only the layer of input nodes and a single layer of neurons constituting the output layer then they are called single layer network. If there are one or more hidden layers, such networks are called multilayer networks. The structures, in which connections to the neurons of the same layer or to the previous layers are allowed, are called **recurrent** networks.

Figure 2: Artificial Neural Network

III. BACKPROPAGATION LEARNING ALGORITHM

The backpropagation algorithm falls into the general category of gradient descent algorithms, which intend to find the minima/maxima of a function by iteratively moving in the direction of the negative of the slope of the function to be minimized/maximized. The main objective is to minimize the error function. The average error function to be minimized (error density) can be given by

$$\mathcal{E}_{av} = \frac{1}{N} \sum_{n=1}^{N} \mathcal{E}(n)$$

In this algorithm, the weights are updated on a pattern-by-pattern basis until one complete epoch has been dealt with. The adjustments to the weights are made in accordance with the respective errors computed for each pattern presented to the network. The arithmetic average of these individual weights over the entire training set is an estimate of the true change that would result from the modification of the weights based on the error function.

A gradient descent strategy is adopted to minimize the error. The chain rule for differentiation turns out to be

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

This can be simplified into

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

The final rule for updating weights becomes

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

where,

$$\delta_j(n) = -\frac{\partial \mathcal{E}(n)}{\partial v_j(n)}$$

$$= -\frac{\partial \mathcal{E}(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)}$$

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

for the last layer and

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

for the intermediate hidden layers.

We use the *batch learning* scheme for weight updating – all the training samples are fed into the network and the change in all weights is computed from each input sample. Then at the end we update the weights according to the sum of all updates. One iteration of inputting all the training samples is called one epoch.

For practical reasons, ANNs implementing the backpropagation algorithm do not have too many layers, since the time for training the networks grows exponentially. Also, there are refinements to the backpropagation algorithm which allow a faster learning.

Hence, the above algorithm can be used to train an Artificial Neural Network (ANN) given the training data and the learning rate. The above network can in general have an arbitrary number of hidden layers and an arbitrary number of hidden neurons in each layer. The number of input layer neurons is decided by the number of input features in each pattern, and the number of output layer neurons is decided by the number of output features in the target values.

There are a few disadvantages associated with backpropagation learning as well:
- The convergence obtained from backpropagation learning is very slow.
- The convergence in backpropagation learning is not guaranteed.
- The result may generally converge to any local minimum on the error surface, since stochastic gradient descent exists on a surface which is not flat.
- Backpropagation learning requires input scaling or normalization.
- Backpropagation requires the activation function used by the neurons to be differentiable.

IV. ANN MODEL FEATURES

A. Activation Function

The user also has an option of three activation functions for the neurons:
- Unipolar sigmoid:

$$f(x) = \frac{1}{1 + e^{-\lambda x}}$$

- Bipolar sigmoid:

$$f(x) = \frac{2}{1 + e^{-\lambda x}} - 1$$

- Tan hyperbolic:

$$f(x) = \frac{e^{\lambda x} - e^{-\lambda x}}{e^{\lambda x} + e^{-\lambda x}}$$

- Radial basis function

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma}e^{-(x-\mu)^2/2\sigma^2}$$

The activation function used is common to all the neurons in the ANN.

B. Hidden Layers and Nodes

The artificial neural network we train for the prediction of image and stock data has an arbitrary number of hidden layers, and arbitrary number of hidden nodes in each layer, both of which the user decides during run-time.

C. Data Normalization

The data is normalized before being input to the ANN. The input vectors of the training data are normalized such that all the features are zero-mean and unit variance. The target values are normalized such that if the activation function is Unipolar sigmoid, they are normalized to a value between 0 and 1 (since these are the minimum and maximum values of the activation function, and hence the output of the ANN), and if the activation function is Bipolar sigmoid or Tan hyperbolic, they are normalized to a value between -1 and 1 and 0 and $1/\sqrt{2\pi}\sigma$ when the activation function is RBF.

The test data vector is again scaled by the same factors with which the training data was normalized. The output value from the ANN for this test vector is also scaled back with the same factor as the target values for the training data.

D. Stopping Criterion

The rate of convergence for the backpropagation algorithm can be controlled by the learning rate η. A larger value of η would ensure faster convergence, however it may cause the algorithm to oscillate around the minima, whereas a smaller value of η would cause the convergence to be very slow.

We need to have some stopping criterion for the algorithm as well, to ensure that it does not run forever. For our experiments, we use a three-fold stopping criterion. The backpropagation algorithm stops if any of the following conditions are met:
- The change in error from one iteration to the next falls below a threshold that the user can set.
- The error value begins to increase. There is a relaxation factor here as well that allows minimal

increase as it is also observed that the error tends to increase by small amount and then decrease again.

- If the number of iterations (or epochs) goes beyond a certain limit. In our case the limit is set to 200.

E. Error Calculation

The error for convergence is calculated as the rms error between the target values and the actual outputs. We use the same error to report the performance of the algorithm on the test set.

F. Cross-validation set

In our algorithm, we give an option of using a crossvalidation set to measure the error of the backpropagation algorithm after every iteration. The crossvalidation set is independent of the training set and helps in a more general measure of the error and gives better results.

V. DATA PRE-PROCESSING

The Nifty Sensex data contained the date, time of day, opening price, closing price, high, low and fractional change in price from the previous time step. Out of these, only four attributes were taken into account, the opening price, closing price, high price and the low price. The output consisted of a single attribute, the closing value. The data was further divided into 60% for training and 40 % for testing the data. Out of the 60 % for training, 40 % was used for just training the model and the rest 20 % for cross validation of the model, wherein whilst the model was being formed, the error was also being computed simultaneously.

Combination of a few samples of the data into one single vector is to be input to the ANN. This introduces some latency to the system. A moving window is used to combine the data points as follows:

$$O_n = (O_{n-k+1}, O_{n-k+2}, \dots O_n)$$

Here k is the latency or the number of previous observations used to predict the next value. Target values are basically the data vector values at the next time step:

$$T_n = O_{n+1}$$

Before being put as input into the ANN, the inputs were normalized according to the '*zscore*' function defined in MATLAB, wherein the mean was subtracted and the value divided by the variance of the data. The target outputs were also normalized according to the target functions, dividing by their maximum values, keeping in mind the upper and lower limits for the respective activation functions ((0,1) for unipolar sigmoid, (-1, 1) for the bipolar sigmoid and the tan hyperbolic functions).

VI. RESULTS AND SIMULATION

The variables that are to be decided by the user are –

- Inputs (Open, Low, High, Close)
- Outputs (Close value)
- Percentage of training data (60%)
- Percentage of testing data (40%)

- Number of previous data points considered for training (5)
- Learning rate, η (0.001)
- Number of hidden layers, nH (2)
- Number of nodes in each hidden layer, n (100,50)
- Maximum number of epochs (30)
- Activation function (Unipolar Sigmoid)
- Value of the activation function parameter (0.5)
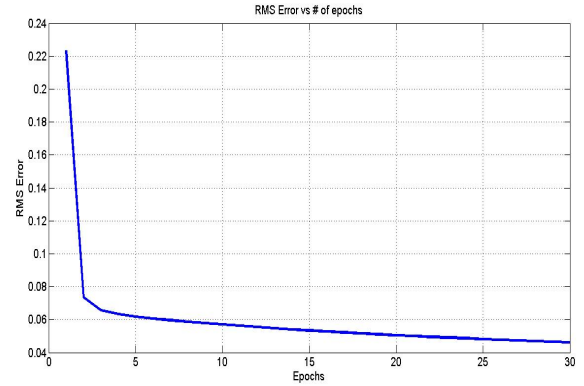
The bracketed values indicate the default values.

Figure 3: RMS Error vs Epochs

The convergence of the RMS error was very smooth as the number of epochs increased. **The best accuracy was achieved in the default case of 96%. The average accuracy over all the cases was 88%.**

The various parameters to be varied by the user were systematically varied by the user, one by one to get the ideal set of parameters so as to obtain the maximum efficiency in the data prediction.

The prediction was carried out using different activation functions (1: Unipolar Sigmoid, 2: Bipolar Sigmoid, 3: Tan Hyperbolic, 4: Radial Basis Functions), keeping all the other parameters at their default values.
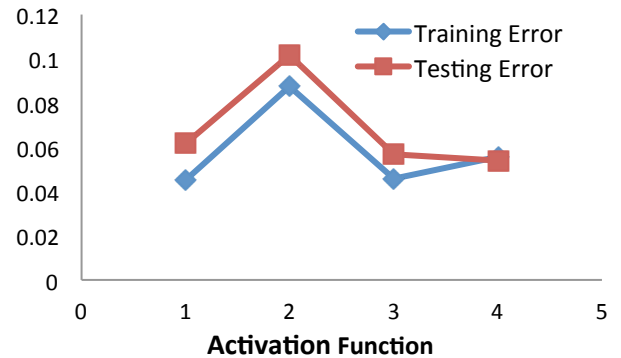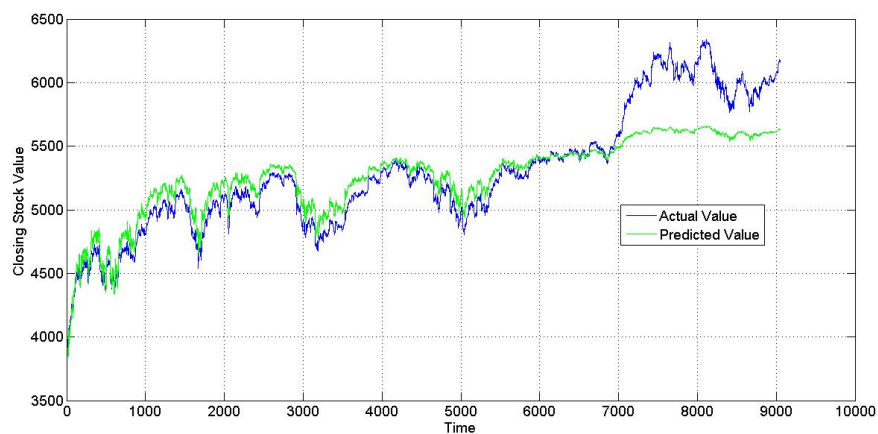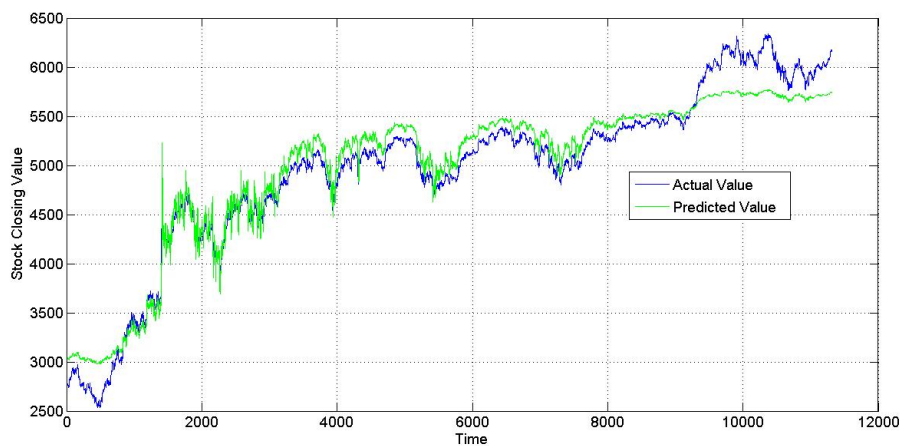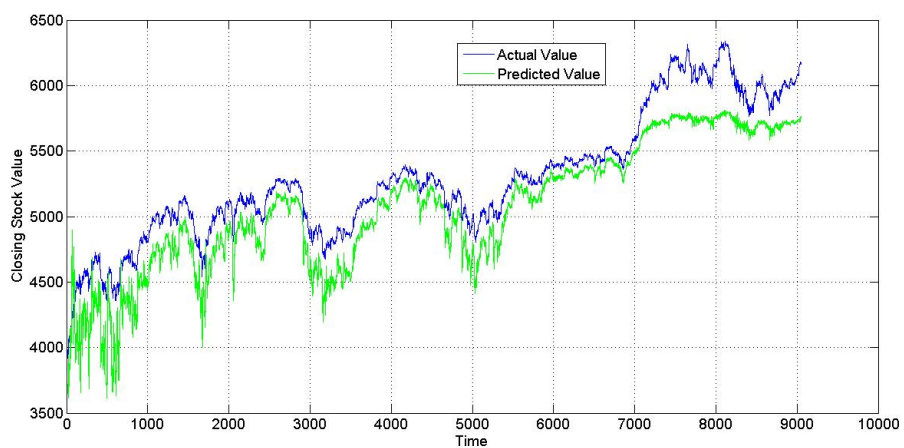

Figure 4: Training and Testing error vs. AF

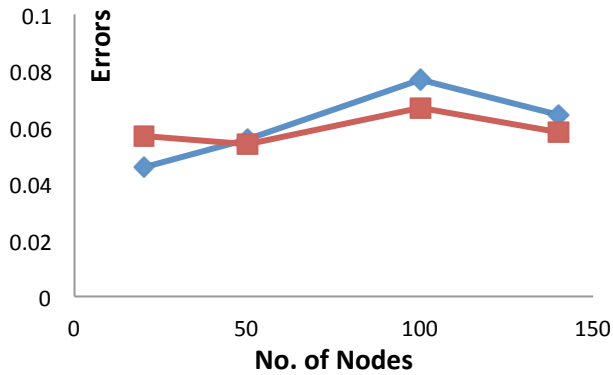**Figure 5: Actual and Predicted Stock Value for single hidden layer**

Figure 6: Actual and Predicted Stock values for 2 hidden layers
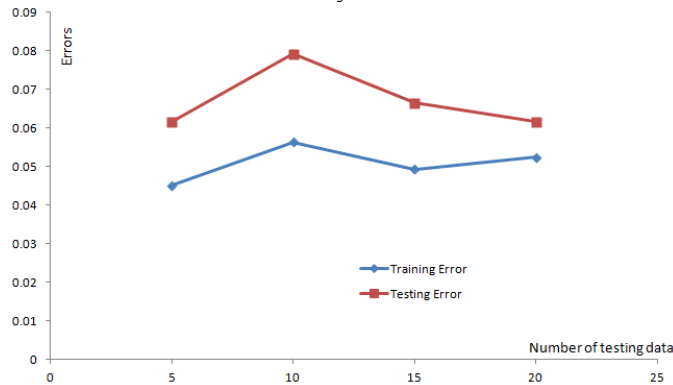

Figure 7: Actual and Predicted Stock Values for 3 hidden layers

Next, the performance was investigated for a single hidden layer, varying the number of layers in the hidden layer.

Figure 8: Training and Testing Errors vs. number of nodes

The number of previous data sets used for testing the data was varied for the double hidden layered neural network.

Figure 9: Training and Testing error vs. number of previous observations

VII. CONCLUSION

In this paper, we described the application of Artificial Neural Networks to the task of stock index prediction. We described the theory behind ANNs and our Neural Network model and its salient features.

The results obtained in both the cases were fairly accurate. As is evident from Fig. 5,6,7, the prediction is fairly accurate unless there is a huge and sudden variation in the actual data like in the right extreme, where it becomes impossible to exactly predict the changes. On the other hand, this also proves the hypothesis that stock markets are actually unpredictable. The minimum error over the testing and the training data was as low as 3.5 % for the case of a single hidden layer.

Thus we can see that Neural Networks are an effective tool for stock market prediction and can be used on real world datasets like the Nifty dataset.

VIII. REFERENCES

[1] J.E. Dayhoff, J.M. DeLeo, "Artificial neural networks: Opening the black box". *Cancer, 91 (8 Suppl.)* (2001), pp. 1615‑1635.

[2] McCulloch, W. and Pitts, W. (1943). "A logical calculus of the ideas immanent in nervous activity". *Bulletin of Mathematical Biophysics*, 7:115 - 133.

[3] Herculano-Houzel, Suzana. "The human brain in numbers: a linearly scaled-up primate brain". *Frontiers In Human Neuroscience*.doi:10.3389/neuro.09.031.2009.

[4] T. Kimoto, K. Asakawa,M. Yoda,M. Takeoka. "Stockmarket prediction systemwithmodular neural network". *Proceedings of the International Joint Conference on Neural Networks*, 1990, pages 1–6

[5] H. Mizuno, M. Kosaka, H. Yajima, N. Komoda. "Application of Neural Network to Technical Analysis of Stock Market Prediction". *Studies in Informatic and Control*, Vol.7, No.3, 1998, pages.111–120.

[6] Beale, R. and Jackson, T.Neural Computing: An Introduction. *Hilger*, Philadelphia, PA, 1991.

[7] W. Zhou, A. C. Bovik, H. R. Sheikh, and E. P. A. S. E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *Image Processing, IEEE Transactions on*, vol. 13, pp. 600-612, 2004

[8] B.G. Malkiel. "A Random Walk Down Wall Street" (6th ed.). *W.W. Norton & Company, Inc.*, 1973