Future Image 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 future images in an image sequence. 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. Principal Component Analysis and Isomap algorithms are used to reduce the dimensionality of feature vectors. We finally test our algorithm on two datasets – the fighter plane dataset and the NASA dataset. The results obtained are acceptable and are discussed in the paper.

Index Terms--Artificial neural networks, Image sequence analysis, Multi-layer neural network, Prediction methods

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.
The artificial neuron given in this figure has \( N \) input, denoted as \( u_1, u_2, \ldots, u_N \). Each line connecting these inputs to the neuron is assigned a weight, which are denoted as \( w_1, w_2, \ldots, w_N \) respectively. Weights in the artificial model correspond to the synaptic connections in biological neurons. The threshold in artificial neuron is usually represented by \( \theta \) 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:

\[
a = \sum_{j=0}^{N} w_j u_j + \theta
\]

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 of the neuron, the formulas are modified as follows:

\[a_i = \sum_{j=1}^{N} w_{ij} x_j + \theta_i\]

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.

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

\[
\mathcal{E}_w = \frac{1}{N} \sum_{n=1}^{N} \mathcal{E}(n)
\]

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

\[
\frac{\delta \mathcal{E}(n)}{\delta w_{ij}(n)} = \frac{\delta \mathcal{E}(n)}{\delta e_i(n)} \frac{\delta e_i(n)}{\delta y_j(n)} \frac{\delta y_j(n)}{\delta w_{ij}(n)}
\]

This can be simplified into
The final rule for updating weights becomes

\[
\frac{\delta^j(n)}{\partial w_{kj}(n)} = -c_j(n)\varphi_j'(v_j(n))y_j(n)
\]

The final rule for updating weights becomes

\[
\Delta w_{kj}(n) = \eta \delta_j(n)y_j(n)
\]

where,

\[
\delta_j(n) = \frac{\delta E(n)}{\partial y_j(n)}
\]

\[
= \frac{\delta E(n) \partial e_j(n) \partial y_j(n)}{\partial y_j(n) \partial y_j(n)}
\]

\[
= e_j(n)\varphi_j'(v_j(n))
\]

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, 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 \(1/2\pi\sigma\) 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 \(\eta\). A larger value of \(\eta\) would ensure faster convergence, however it may cause the algorithm to oscillate around the minima, whereas a smaller value of \(\eta\) 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.

In addition we also use the mean structural similarity (MSSIM) index [7] to evaluate the performance of the algorithm on the image data. In MSSIM, the structural information is defined as attributes that represent the structure of the objects in the scene, independent of average luminance, and contrast. Therefore, MSSIM has been adapted as one the image similarity measures for predictive applications. MSSIM index for a colour image is found by averaging the MSSIM indices for its R-image, G-image and B-image separately as given by the expressions:

\[
\text{MSSIM}(X(t), Y(t)) = \frac{1}{H} \sum_{t=1}^{H} \text{SSIM}(X_i(t), Y_i(t))
\]

\[
\text{SSIM}(X_i(t), Y_i(t)) = \left( \frac{2\mu_{X_i(t)}\mu_{Y_i(t)} + C_1}{\mu^2_{X_i(t)} + \mu^2_{Y_i(t)} + C_1} \right) \left( \frac{2\sigma_{X_i(t)Y_i(t)} + C_2}{\sigma^2_{X_i(t)} + \sigma^2_{Y_i(t)} + C_2} \right)
\]

In the above equation, \(C_1 = (K_1L)^2\) and \(C_2 = (K_2L)^2\). Here \(\mu_{X_i(t)}\) and \(\mu_{Y_i(t)}\) are the mean instances of the \(h\)th window, \(H\) is the total number of windows and \(\sigma_{X_i(t)Y_i(t)}\) is the covariance used as a measure of contrast. Here \(L\) is the dynamic range of pixel values (255 for 8 bit image) and \(K_1, K_2\) are small constants. We take the average of the MSSIM for all three R,G,B components.

### F. Cross-validation set

In our algorithm, we give an option of using a cross-validation 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

Two datasets for image sequences were available for this project. One of them called the _fighter_ dataset contained a sequence of images of a fighter plane landing on a naval station. It had a total of 200 images, out of which 194 were used for training and 6 for testing.

The other dataset available, called _NASA_ dataset, showed images of a moon in the sky moving across a planet. It had a total of 160 images, out of which 147 were used for training and 13 were used for testing.

The pre-processing was similar for both the datasets, as explained below:

- The dataset was divided into three subsets by separating the R,G and B channels of each image.
- We reduce the number of dimensions in each subset using Principal Components Analysis (PCA) and Isomap algorithms. Typically the dimensions are reduced to 100 in each case. PCA and Isomap are explained in detail in the further subsection.
- Using a given number of previous images, we form sequences of input data vectors in a manner similar to that for Sensex data. Input data vectors into the consist \(k\) images concatenated one after the other:
  \[O_n = (O_{n-k+1}, O_{n-k+2}, \ldots, O_n)\]
- The target value for each input is simply the next image in the sequence:
  \[T_n = O_{n+1}\]
- We normalize the input vectors and the output targets to according to the data normalization scheme described in the previous section and used for Sensex data.

After the above pre-processing we train three ANNs – one for each R,G and B channels. During prediction, we input a sequence of \(k\) images into the ANN, after applying PCA and normalizing the data. The output obtained corresponds to the predicted image. We rescale this output to remove the effect of normalizing, and then invert the PCA process to obtain an image of the same dimensions as the original one. Then we combine the three R,G and B channels to obtain the final predicted output from the algorithm. This is then compared with the original next image in the sequence.

#### A. PCA: Principal Component Analysis

Principal Component analysis of data is a dimensionality reduction technique wherein the data dimension is reduced by mapping it into its eigen-space. In the process the top-k eigenvectors are chosen to reflect the directions of maximum variability of the data. In practice, the lower dimensional representation is calculated by a singular value decomposition of the data followed by extraction of its top-k eigenvectors (where k is chosen by the amount of energy to be retained) and mapping to the top-k eigenvector space.

#### B. Isomap

Isomap – short for isometric feature mapping – was one of the first algorithms introduced for manifold learning. The algorithm is perhaps the best known and most applied among the multitude of procedures now available for the problem at hand. It may be viewed as an extension to Multidimensional Scaling (MDS), a classical method for embedding dissimilarity information into Euclidean space. It consists mainly of 2 steps:

1. Estimate the geodesic distances (distances along a manifold) between points in the input using shortest-path distances on the data set’s k-nearest neighbor graph.
2. Use MDS to find points in low-dimensional Euclidean space whose inter-point distances match the distances found in step 1.

This ensures that the geodesic distances in the data are preserved when they are mapped to lower dimensions. In
contrast to PCA, Isomap represents a non-linear dimensionality reduction algorithm. This was tried on the datasets as they involved movement of objects that might not be well abstracted by PCA in lower dimensions but by Isomap due to its non-linearity.

VI. RESULTS AND SIMULATION

The image data consists of two data sets, the NASA data set (that shows the moon against the sky above the surface of a planet) and the fighter data (that shows a fighter plane landing on a naval sea base).

Each of these were analyzed in two modes, the RGB mode and the grayscale mode and the attribute reduction was done using two algorithms, the Principal Components Analysis (PCA) and the Isomap technique.

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

- PCA Dimension (100)
- Isomap Dimension (3)
- Number of previous data points considered for training (3)
- Learning rate, $\eta$ (0.001)
- Number of hidden layers, $n_H$ (2)
- Number of nodes in each hidden layer, $n$ (300,100)
- Maximum number of epochs (300)
- Activation function (Unipolar Sigmoid)
- Value of the activation function parameter (0.5)

The rms error again shows a very smooth convergence with increasing number of epochs.

![Figure 3: RMS error vs. Epochs](image3)

The error was measured in terms of the MSSIM index. Varying the number of previous images used for the testing module, the MSSIM index followed the pattern where the maximum similarity between the predicted output and the target was achieved for testing with 8 previous images.

![Figure 4: Average MSSIM vs. Number of previous observations used to predict image](image4)

A. Fighter Dataset

1) Grayscale images using PCA

![Original Image](image_original)![Predicted Image](image_predicted)

2) RGB images using PCA

![Original Image](image_original)![Predicted Image](image_predicted)

B. NASA Dataset

1) RGB Images using PCA

![Original Image](image_original)![Predicted Image](image_predicted)
The accuracy is noticeably higher in the NASA dataset as compared to the fighter data. This is because we are reducing the dimension of the picture from tens of thousands of pixels to about a hundred, choosing only the most salient ones. So when there is a higher degree of complicacy in the image, like in the fighter set, the program is not able to accurately predict the entire picture (The MSSIM index averages at 84% for the NASA data and 55% for the fighter data). Also, the grayscale mode is slightly more accurate than the RGB mode, because in the RGB mode, the pixels are broken down into R, G and B components, each of them predicted separately and then recombined to give the final output whereas the grayscale model needs to deal with only a single intensity value. The third point to be noted is that the PCA gives a better result in terms of the MSSIM index but the most prominent feature is not very clear, as compared to the Isomap technique, which is evident from the moon in the NASA data.

VII. Conclusions

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

The results obtained in both the cases are acceptable. It is to be noted that we are reducing the dimensionality of the data from an order of $10^4$ to at most 100. The quality of images reconstructed from this low dimensional data is notably nice and is a strength of our algorithm. The Isomap algorithm doesn’t work as well as hypothesized perhaps due to the very low dimension of embedding. Higher dimensions lead to exponentially more space and computation in Isomaps. Our algorithm provides a generic framework to test on any image sequence. The similarity indices obtained are good for certain cases (with one case in PCA, we obtained a similarity of 92%).

Thus we can see that Neural Networks are an effective tool for future image prediction but work needs to be done to make them applicable in the real world scenario.

VIII. References