Arm Movement Prediction Using Neural Networks

Fred Stakem and Ghassan AlRegib
School of Electrical and
Computer Engineering
Georgia Institute of Technology
Savannnah, Georgia 31407
Email: fstakem@gatech.edu, gregib@ece.gatech.edu

***Abstract*—Whether interacting with a Collaborative Virtual Environment, or CVE, locally or one networked across the Internet, any delay in the system can lead to a reduced sense of immersion. Input sensor delay and network delay are two common problems in CVE design that can be overcome with the application of prediction algorithms to the system. The purpose of this experiment was to assess the quality of feed forward back propagation neural networks in predicting natural avatar arm movement typically used in a CVE. In addition the experiment attempts to find the bounds for precise neural network prediction. The results show many different combinations of back propagation neural network topologies are capable of predicting up to 400 ms of human arm movements relatively accurately.**

I. Introduction

Interacting with others using a CVE can be done with a multitude of different equipment, but the most common methods generally employ sophisticated input equipment such as data gloves and trackers. One major problem with CVEs is the inherent delay caused by the input devices and the network when systems are interconnected. Delay hampers natural interaction and reduces the immersion of the application [1], [2]. One method to overcome the delay is to use predictive algorithms to estimate the future state from the present state and past states. As long as the predicted state error is relatively small, prediction techniques can be used for both local and remote users.

Some of the same difficulties, such as excess delay, that arise in networking CVEs arise in the networking of 3D video games. Dead reckoning is a technique used in video gaming and Distributive Virtual Environments, DVEs, to combat the effects of delay while reducing network bandwidth [3]. The dead reckoning algorithm transmits an entity's position information along with its velocity and sometimes acceleration. The velocity and potentially acceleration are used to predict the future position of an entity on networked hosts. As long as the entity stays within a specific error bound no new state information needs to be transmitted. The networked host will simply continue predicting the entity's next state until new information is received. Although dead reckoning has been applied to DVEs for years, the ability of the algorithm to predict human movement remains to be proven.

Another common method used to predict the future state of a system is the use of neural networks [4]. Although better known for pattern matching neural networks can and are used for time series prediction with the standard techniques being well documented [5], [6]. The most difficult part of using a neural network is not their programming, since many software libraries exist, but the collection of training data and the estimation of system parameters. First a training set must be collected on which the neural network is to be trained. Once the data is collected it must be properly parsed and conditioned to allow the neural network to perform optimally. When the data is ready for the neural network a multitude of parameters must be defined for the network, which is often likened more to an art than a science. Finally the training data is used to train the neural network to minimize the error for inputs with similar properties.

One of the most common and well understood types of neural networks is the feed forward back propagation network which can also be applied to time series analysis and prediction [7]. When a back propagation neural network is used for prediction the difference is not in its topology or connectivity but in the input and output signals. Instead of the input and output nodes representing the signal at one particular time instance, they represent the signal at various time shifted samples. To use a neural network for prediction the input feed into the neural network consists of the present state sample and a number of previous state samples as can be seen in figure 1. Each sample is input to one input node so that the signal from multiple previous sample times is input at the same instance. Therefore the number of state samples used for prediction dictates the number of input nodes the neural network needs. As with the input each of the output nodes from the neural network represents a time shifted version of the signal except the output is the future estimate. So the number of samples the neural network needs to predict ahead dictates the number of output nodes the network needs.

The sampling rate is the most important characteristic of the input signal that must be set correctly. Generally this implies down sampling the input to a rate that allows the neural network to perform optimally. The reason for down sampling is that an oversampled signal will have redundant data not essential for state prediction which behaves like noise to the neural network. Redundant data requires a larger input pattern for the same degree of information requiring a larger and more complex neural network for the same level of prediction. By calculating and plotting the mutual information of the input signal at different sampling rates, the minimum of the mutual information can more easily be found. The minimum of the

978-1-4244-2390-3/08/$25.00 ©2008 IEEE

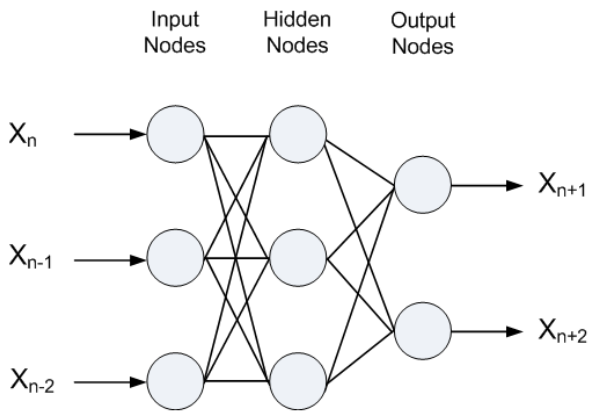


Fig. 1. Example neural network with three input nodes, three hidden nodes, and two output nodes.

Fig. 2. Subject using the CVE for a collaborative task.

mutual information corresponds to the sampling rate where the least amount of redundant data is contained in the signal and should be used as the new sampling rate of the signal. This technique can easily be done by visually inspecting the mutual information plot.

Once the input signal is properly sampled the neural network must be designed. The number of input nodes corresponds to the number of time shifted samples used for prediction while the number of output nodes corresponds to the number of samples predicted. Dynamical systems theory can be used as a guide when selecting the optimal number of input nodes to be used for quality prediction. One useful technique in dynamical systems theory is to calculate the percent of false nearest neighbors for different numbers of inputs, or the embedding dimension, of the system. A value where the percent of false nearest neighbors stabilizes indicates a suitable point for designing the neural network and is a good guide for experimentation. Although the theory is fundamentally sound it does not always lead to the optimal solution making it up to the designer to test and ultimately select an appropriate value.

II. Data Collection

Typically only one set of data is collected for a neural network. The data is then split between a set used for training and a set used for testing. It is imperative that the data collected is specific enough to allow precise prediction of the movements, but general enough to allow many different types of movements to be predicted. Previous research in arm movements used in CVEs showed many different types of movements have a similar Gaussian speed profile with the variation being the height and width of the Gaussian shape [8]. From this knowledge a simple experiment was devised to collect data that would reflect typical CVE movement patterns with a wide degree of different Gaussian shapes.

The virtual environment for the experiment consisted of a small virtual room with a central table in which user interaction was designed to take place. 5DT data gloves and Nest of Birds trackers were used as input devices translating the subject's hand and arm movements into appropriate avatar movements. Instead of having a multitude of different types of objects to interact with, virtual blocks were used for the collaborative tasks. The virtual blocks sat on top of the table and allowed subjects to build many different shapes with them. Although the virtual blocks require subjects to perform dexterous manipulations, they are generic enough so that they do not require object specific movements such as a wrench would.

To collect the data a human subject was asked to repeatedly perform an experiment while their movements were recorded (figure 2). The purpose of the experiment was for the subject to use the virtual environment to stack three randomly placed blocks on top of a target in the center of the table top. At the start of each experiment the subject placed their right hand on the target in the center of the table. Once the subject placed their hand on the target three blocks were randomly placed on the table top. The subject was instructed not to rush, but to stack the three blocks on the target where their right hand started. Since the experiment was not time based its intent was on capturing the majority of natural movements used in the performance of common tasks.

The task the user was asked to perform was not arbitrary but specifically designed to capture many different types of movements. Previous research has identified reaching and manipulative movements as common movements used in task performance in a CVE. By making the subject stack blocks placed randomly about the table the subject must perform both reaching and manipulative movements to accomplish their goal. In addition the random distance of the blocks from the center of the table insured that both short and long reaching movements were required. Although the subject was not required to handle any complex virtual objects, the act of stacking small blocks required a great deal of dexterous manipulation thus recording intricate movements as well. The overall goal was accomplished by having the subject perform many different movements in a natural non scripted fashion without a large number of object specific movements.

In order to collect a sufficient amount of data the subject performed the experiment 100 times over the course of three different days. Half of the sets were designated for training

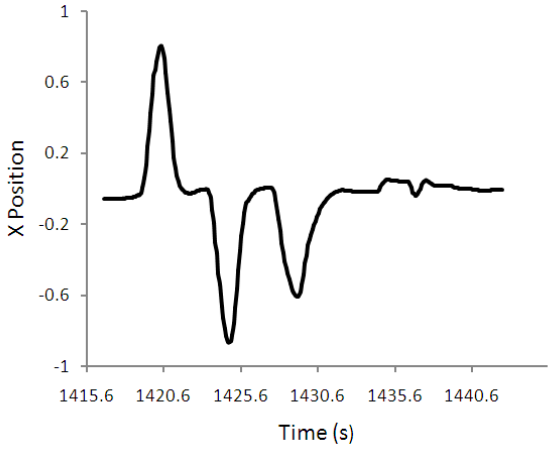


Fig. 3. Typical arm movement from experiment.

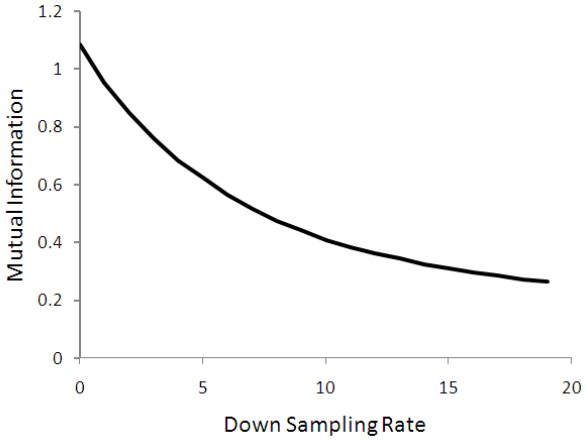


Fig. 4. Mutual information of the data sets.

and the other half for testing. Each set contained roughly three reaching movements and many small manipulative movements. The reaching movements clearly show up on the time series graphs as large Gaussian shaped curves, while the more subtle manipulative movements are more difficult to discern. A closer inspection of the data does however show the lower amplitude but still roughly Gaussian shape of the smaller movements. The typical length of a data set was approximately 20 seconds and was sampled at 100 ms intervals. Since the three different axes exhibited similar patterns of movement only one axis was utilized for a more in depth analysis. It is envisioned that each axis would have a separate neural network trained on its data so that motion prediction would be done separately for each axis. A typical sample can be seen in figure 3.

III. Neural Network Design

Before the neural network was trained, the data was analyzed using dynamical system theory. Dynamical systems theory provides the theoretical framework for non linear time series prediction that back propagation neural networks implement. By first analyzing the data a better understanding of what needed to be input to the neural network was made. First the mutual information versus sampling rate for each data set was found using the TISEAN software package [9]. Since the mutual information curves tended to have a similar shape and similar values the mean value at each sampling rate was taken as a good estimate. This condensed all of the data curves to a single curve. The mutual information was plotted versus the sampling rate and is shown in figure 4. No clear minimum value was identified in the condensed data or individual data sets as all curves show a continuously decreasing slope. Without a distinct minimum the data does not appear to need down sampling, and therefore no down sampling was done.

Next the embedding dimension was estimated using the false nearest neighbor test. By finding a stable embedding dimension a good estimate of the number of inputs needed for state prediction can be found. As with the previous mutual information test, this was done using the TISEAN software package. The results from the individual data sets were similar indicating no anomalies in the data. Each set of data was analyzed to find the best embedding dimension. The data was inconclusive since with the data suggesting a very large embedding dimension of ten or higher. A more conservative value of five has a greatly reduced amount of false nearest neighbors and seems more appropriate given the sampling rate.

Once the ideal sampling rate and embedding dimension were found the remainder of the neural network parameters were chosen. Using many of the standard or recommended values for feed forward back propagation neural networks greatly simplified this task. The three main parameters left to be defined were the number of input, hidden, and output nodes. Instead of blindly selecting these values or trusting the theoretical results it was determined that testing multiple neural networks with different parameters would lead to the optimal solution, or set of solutions. At first selecting the number of hidden nodes seemed like a difficult choice since the number used varies vastly from researcher to researcher. A comprehensive analysis done by G. Zhang et al. comparing the results of many research papers suggests that on average the optimal number of hidden nodes is the same number of inputs nodes in the network [10]. It was therefore decided to reduce the number of networks tested by directly tying the number of hidden nodes in the network to the number of input nodes. In addition to reducing the number of networks tested, this greatly simplified the task of finding the optimal number of hidden nodes.

Instead of defining a single value for the number of input and output nodes a range of values was chosen. The number of input nodes was varied from three to seven so the theoretical embedding dimension of five was the center point of the tested values. For each different number of input nodes the number of output nodes was varied from one to ten. A value of ten was chosen as the maximum predicted sample since given the sampling rate this would correspond to a whole second of

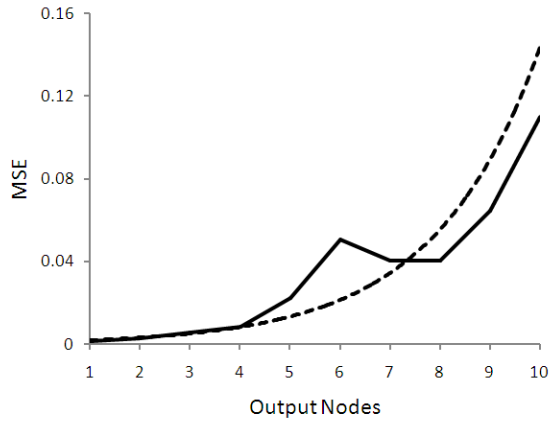


Fig. 5. Quality of the neural networks with trend line.

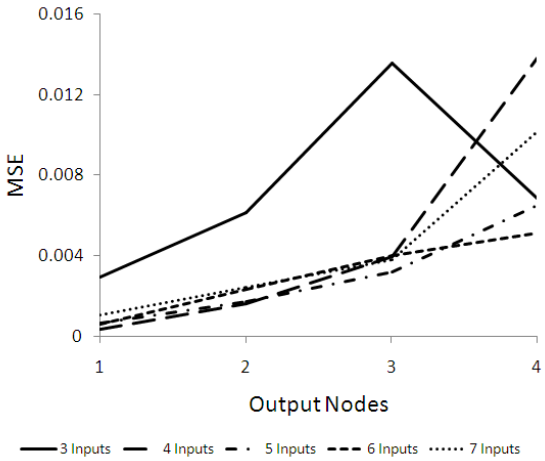


Fig. 6. Quality of the most accurate neural networks.

prediction. By setting the number of hidden nodes to the same value as the number of inputs nodes this produced a set of 50 different neural networks to be tested.

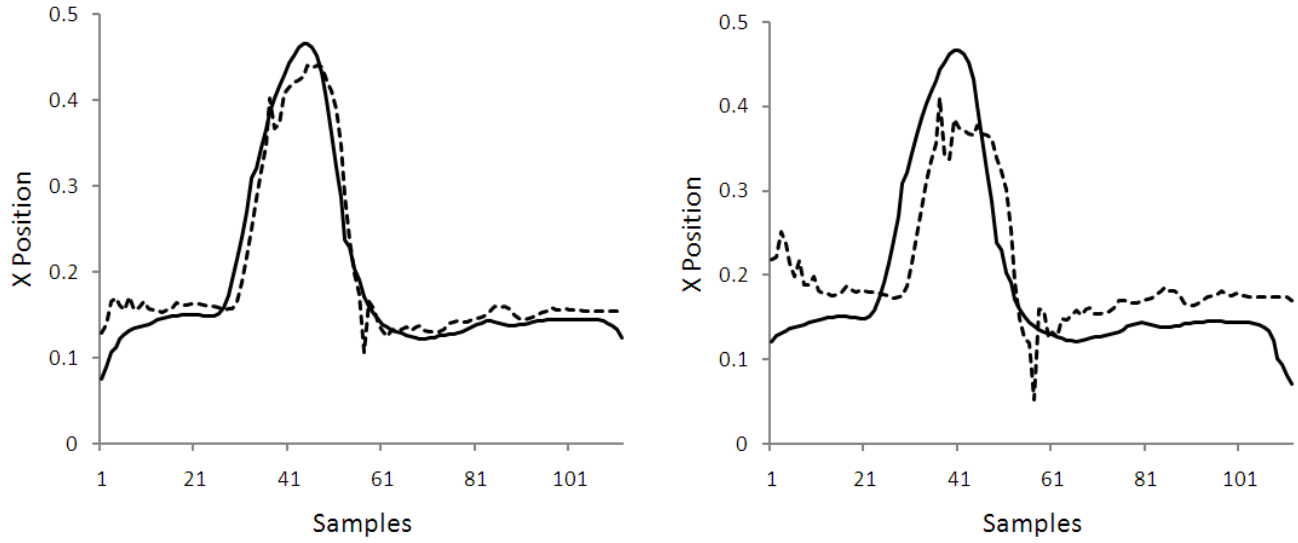
Finally after the neural network parameters were set the neural network was trained with the 50 training sets. Although continuously training each neural network seems like it would lead to the optimal solution, this is not always the case. In some cases over training a network can lead to a neural network that only works well for the training data. To alleviate this problem the neural networks were tested at intervals throughout the training process with data not in the training set. Of the 50 testing data sets the first 25 were reserved for this purpose so that 75 data sets were actually used in the training process. After each training interval one of the first 25 testing sets were chosen at random and tested on the network returning the MSE from the test set. If the MSE of the testing set was lower than or equal to the previously recorded MSE then training continued and the MSE was saved. If the MSE of the testing set was larger than the previously recorded MSE then training stopped. Once training had stopped the networks training was considered complete and the networks were saved for later testing.

IV. Results

After the neural networks were trained the remaining 25 data sets not used in the training process were tested on the 50 different neural networks. The MSE for each test was recorded and the mean found for each network. This gave a snapshot of the quality of each network which was utilized in an attempt to find the optimal solution. To properly analyze the data a graph was needed to illustrate the quality, given by the MSE, versus the amount of prediction. By grouping the neural networks with the same number of input nodes together into a curve the data was condensed into five curves. Except for a few spurious data points, possibly caused by the small size of the data set, all of the curves had a similar exponential growing shape. To better illustrate this trend the mean of the five curves was taken and plotted in figure 5. The amount of prediction is in samples with each sample representing 100 ms. An exponential trend can be seen in the data shown by the dotted line in the figure. Prediction for a few samples ahead of time is quite accurate, but the error increases rapidly the further ahead the network attempts to predict.

Focusing in on the more linear region with shorter amounts of prediction, figure 6 plots the five different curves corresponding to the five different numbers of input nodes. It is difficult to observe any trend in the data relating to the number of input nodes. There is decreasing accuracy the more samples ahead predicted, but the number of input nodes does not seem to make a difference. Only the curve for three inputs nodes deviates from the others suggesting that any network with a number of input nodes slightly greater than three will give similar results. The befuddling nature of the results could be attributed to the small data set or possibly some characteristics of the input signal. Not only did the final results show no clear optimal number of inputs, but the false nearest neighbor test done on the input signal suggested an extremely high embedding dimension. Al-though the results obtained show neural networks work well more research needs to be done on the optimal embedding dimension.

The clearest way to reveal the quality of a neural network at predicting the future state is to show the prediction estimate for an individual set of data. Showing a complete set of data is too large to effectively demonstrate the difference in prediction quality so a single reaching movement from a data set was chosen. Figure 7 illustrates the prediction quality, shown by the dotted lines, of a neural network with six input nodes and eight output nodes at two different prediction intervals. Plotted are the actual signal and the estimated signal for prediction four and eight samples ahead. The graphs clearly shows the increasing degradation of the estimated signal as the neural network attempted to predict further ahead. One sample ahead prediction is very accurate making it difficult to tell the difference between the actual and the predicted signal. At four samples ahead, or 400 ms of prediction, the shape of the estimated signal closely resembles the actual signal,



(a) reaching prediction with four samples, or 400 ms, of prediction

(b) reaching prediction with eight samples, or 800 ms, of prediction

Fig. 7. Reaching movement prediction and actual signal for a neural network with 6 input, 6 hidden, and 8 output nodes.

but the quality is slightly degraded. Analysis of multiple data sets points to 400 ms as the soft limit of effective prediction, but this may differ depending on the accuracy needed for the application. Finally at eight samples ahead, or 800 ms of prediction, the estimated signal shows excess lag and degradation making estimation difficult if not impossible. The excess lag when attempting greater amounts of prediction suggests prediction is ineffective for large prediction intervals.

V. Conclusions

Although there are many sources of delay in a CVE, proper use of prediction algorithms can mitigate the effects of excess delay. In particular this experiment has shown feed forward back propagation neural networks as an effective solution for human movement prediction in CVEs. Not only can neural networks be used to predict one sample ahead of time, but have been shown to effectively and accurately predict up to four samples, or 400 ms, ahead. Greater amounts of prediction are theoretically possible but have been shown to be ineffective and have excess delay. Since the worst case scenario for many real time networked applications is 200 ms, the amount of prediction neural networks allow is more than ample to resolve some of the worst delay conditions experienced on the Internet. In addition once the neural network is properly trained, state prediction requires very few computations allowing the technique to be used in systems with limited resources such as embedded applications. Since there are many good neural network libraries, especially in low level languages such as C, porting the libraries to an embedded system is not overly complex. The next logical step for the current research is to apply the neural networks to real time applications and judge their quality based on user feedback.

References

[1] K. S. Park and R. Kenyon, "Effects of network characteristics on human performance in a collaborative virtual environment." Houston, TX, USA: Proceedings IEEE Virtual Reality, 1999, pp. 104–11.

[2] I. Vaghi, C. Greenhalgh, and S. Benford, "Coping with inconsistency due to network delays in collaborative virtual environments." London, UK: VRST'99. Proceedings of the ACM Symposium on Virtual Reality Software and Technology, 1999, pp. 42–9.

[3] S. Singhal and M. Zyda, *Networked Virtual Environments: Design and Implementation*. York, New York: ACM Press, 1999.

[4] A. Garrett, M. Aguilar, and Y. Barniv, "A recurrent neural network approach to virtual environment latency reduction," vol. 3. Honolulu, HI, USA: Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02, 2002, pp. 2288–92.

[5] A. Ulbricht, G. Dorffner, and A. Lee, "Forecasting fetal heartbeats with neural networks," vol. 1. London, UK: Solving Engineering Problems with Neural Networks. Proceedings of the International Conference on Engineering Applications of Neural Networks (EANN'96), 1996, pp. 403 – 6.

[6] C. Ulbricht, G. Dorffner, S. Canu, D. Guillemyn, G. Marijuan, J. Olarte, C. Rodriguez, and I. Martin, "Mechanisms for handling sequences with neural networks," *Austrian Res. Inst. Artificial Intelligence*, 1992.

[7] R. Frank, N. Davey, and S. Hunt, "Time series prediction and neural networks," *Journal of Intelligent and Robotic Systems*, vol. vol. 31, pp. 91–103, 2001.

[8] F. Stakem, G. Alregib, and B. Juang, "Towards modeling human arm movement in a cve." Verona, Italy: IMMERSCOM, 2007, 2007.

[9] R. Hegger, H. Kantz, and T. Schreiber, "Practical implementation of nonlinear time series methods: The tisean package," *Chaos*, vol. vol. 9, no. 2, pp. 413–35, 1999.

[10] G. Zhang, B. Patuwo, and M. Hu, "Forecasting with artificial neural networks: the state of the art," *International Journal of Forecasting*, vol. vol. 14, no. 1, pp. 35 – 62, 1998.