Artificial Neural Network-Based Prediction of Human Posture

Mohammad Bataineh, Timothy Marler, and Karim Abdel-Malek

Virtual Soldier Research Program–Center for Computer-Aided Design
The University of Iowa
Iowa City, IA 52242, USA
mohammad-bataineh@uiowa.edu,
{tmarler,amalek}@engineering.uiowa.edu

Abstract. The use of an artificial neural network (ANN) in many practical complicated problems encourages its implementation in the digital human modeling (DHM) world. DHM problems are complicated and need powerful tools like ANN to provide acceptable solutions. Human posture prediction is a DHM field that has been studied thoroughly in recent years. This work focuses on using a general regression neural network (GRNN) for human posture prediction. This type of ANN has advantages over others when incorporated in DHM problems like posture prediction. A new heuristic approach is also presented in this study to determine the GRNN parameters that lead to the best performance and prediction capability. The results are promising: a high success rate is obtained for predicting 41 outputs, which represent the upper-body degrees of freedom of a human model. This work initiates future focus on embedding GRNN to generalize human posture prediction in a task-based manner.

Keywords: Digital human modeling and simulation, artificial neural network, posture prediction.

1 Introduction

Digital human modeling (DHM) is a human representation on computer software or a computer model used to perform analyses and evaluations related to human performance. DHM studies have developed and facilitated the study of many human-related fields. As one of these fields, human posture prediction is incorporated in studying and analyzing many ergonomic studies, human-machine workplaces, vehicle designs, etc. Hence, studying posture prediction is critical for understanding human performance when a powerful method or approach is needed to predict accurate and reliable postures. This reliability can be provided by incorporating prediction tools like an artificial neural network (ANN). ANN is a mathematical model for predicting system performance (i.e., system output) inspired by the structure and function of human biological neural networks. ANNs have been studied extensively and applied in various problems [1] and [2]. Their benefits have not yet been fully realized in the context of human-posture models because there are many variations of ANNs, and selecting the appropriate form and associated parameters for a particular application can often be more of an art than a science.

The applications of solving human posture prediction problems are not new; various approaches, including some types of ANN, have been used to address these problems. A simulation of in-vehicle seated reaching movements was presented with a seven-degree-of-freedom (7-DOF) linkage structure depending on an inverse kinematics approach [3]. Then, an efficient numerical formulation was developed for the prediction of real postures [4]. Moreover, ANN has been used to predict the next steps or postures for many types of postures [5], [6]. This paper presents the use of the general regression neural network (GRNN) type of ANN to predict human posture for the following reasons: 1) to investigate potential issues when using GRNN to simulate tasks that involve contact constraints or other conditions involving Cartesian locations, 2) to provide initial work in posture prediction to discover the limitations of using GRNN in this DHM problem, and 3) to demonstrate the feasibility of predicting a large number of outputs using GRNN.

In general, the current state of the art does not demonstrate the use of ANNs for direct manipulation of joint angles in the context of a complete human model with a large number of DOFs. Most of the applied work was done using traditional ANN types like the feed-forward neural network, which has memory and training limitations when predicting a large number of outputs. Moreover, human posture prediction needs to be studied in a task-based manner like touching a point, box lifting, sitting/standing, etc. Thus, we propose not only exploring the use of ANNs for predicting upper-body posture but doing so in the context of a complete 41-DOF DHM. Furthermore, we demonstrate the determination of the appropriate ANN and ANN parameters for application to human posture prediction. Thus, the following contributions are achieved in this paper: 1) task-based posture predictions for a 41-DOF human model using ANN, 2) the first use of GRNN in task-based posture prediction with highly reasonable results, 3) prediction of a relatively large number of outputs (41) for a 41-DOF human model, and 4) introduction of a new fully automatic strategy for determining the Gaussian width (GW) parameter for optimal network performance.

2 Background

2.1 Human Model

The underlying human model used with this work is Santos [7], which is built on a biomechanically accurate musculoskeletal model with 55 DOFs. Santos's anthropometry can be altered on the fly as can the range of motion (ROM) for each joint. Human posture prediction on Santos is done using an optimization-based approach; conceptually, the joint angles (one for each DOF) provide design variables. The objective function(s) is one or more human performance measures, such as energy, discomfort, joint displacement, etc.

2.2 Neural Networks

ANN involves many neurons arranged in multiple interconnected layers. In much the same way that one learns, ANNs essentially provide a high-dimension surface (representing system output) fit to variables that control system behavior. The process of fitting the hyper surface to data points is called training, and the data points are called grid points. A single grid point represents a set of input parameters used to train the ANN. The term "off-grid points" refers to input parameters that were not actually used in a training case. This study uses GRNN, which is a type of radial-based neural network (RBNN) [8], [9]. The use of GRNN has advantages, including fast training, smooth prediction, and the ability to handle a relatively large number of inputs and outputs.

In the GRNN architecture, $\boldsymbol{x} = [x_1, x_2, \dots, x_R]$ (R is the number of inputs) provides the input for each neuron in the hidden layer. The hidden layer has Q neurons $[1, 2, \dots, Q]$ (Q is the number of training cases). Inside each hidden neuron, there is a radial transfer function that produces output depending on the provided input, so the final output depends on the radial distance of the input from the basis function's center [8]. The Gaussian function is the most popular type of radial function [8], so it is used in this study. The hidden neuron's output enters all neurons in the output layer. Each neuron in the output layer essentially combines the received lines (the outputs of all hidden neurons) in a weighted sum to provide the final network outputs. The output layer has N number of neurons, which is the number of outputs $[y_1, y_2, \dots, y_N]$.

Figure 1 shows a flow chart for the mathematical steps that are calculated inside each neuron in both the hidden and output layers. Once the neuron receives the input **x**, the sum of the absolute values between **x** and the components of the vector $\boldsymbol{W}_i^I$ (Equation 1) is calculated in the "Distance Function" to produce A_i, as in Equation 2. The dimension of the input weight matrix $\boldsymbol{W}^I$, as shown in Equation 3, is QxR. Each row of $\boldsymbol{W}^I$ is referred to as an input weight vector associated with a corresponding hidden neuron. Then, the value A_i is multiplied by the bias constant B in the "scaling function" to provide a_i (Equation 4), which is called the radial distance. The bias B is responsible for the network sensitivity, which is directly calculated from a network parameter (GW). More detail about those two terms will be provided in the method section. The last step is to calculate the radial function outputs $h_i(a_i)$ to provide the neuron's output h_i (Equation 5), which represents the hidden neuron output. $\boldsymbol{h} = [h_1, h_2, h_3, h_Q]$ represents the output from Q hidden neurons, which are provided to each neuron in the output layer. The figure shows the kth output neuron, which provides the kth output y_k (Equation 6). This output is calculated by calculating the sum of the dot product between the provided $\boldsymbol{h}$ and the output weight vector $\boldsymbol{W}_k^O$ (Equation 7) and divided by the sum of $\boldsymbol{h}$ components. In the output weight matrix $\boldsymbol{W}^O$ (Equation 8), $\boldsymbol{W}_k^O$ refers to the kth weight vector associated with a corresponding output neuron.

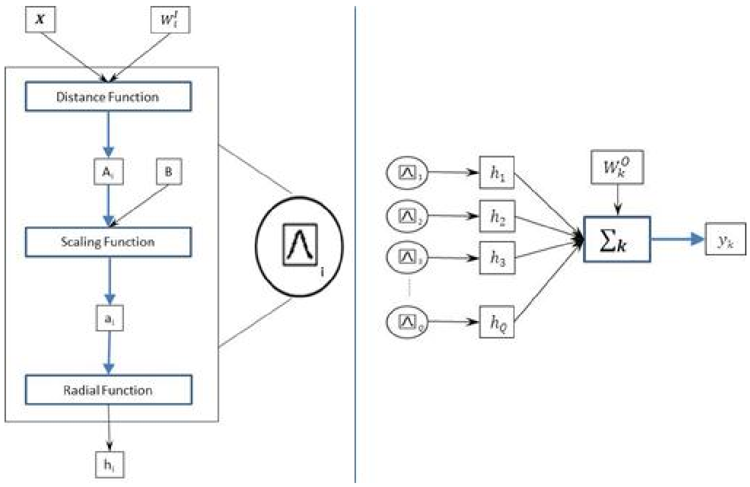

Fig. 1. The ith neuron at the hidden layer (at left), and the neuron at the output layer (at right)

$$\boldsymbol{W}_i^I = [w_{i1}^I \ w_{i2}^I \ldots . w_{iR}^I \ ] \tag{1}$$

$$A_i = \sum_{j=1}^{R} \left| w_{ij}^I - x_j \right| \tag{2}$$

$$\boldsymbol{W}^I = \begin{bmatrix} w_{11}^I w_{12}^I \ldots \ w_{1R}^I \\ w_{21}^I w_{22}^I \ldots \ w_{2R}^I \\ \ldots \ldots \ldots \ldots \ldots \ldots .. \\ w_{Q1}^I w_{Q2}^I \ldots \ w_{QR}^I \end{bmatrix} \tag{3}$$

$$a_i = A_i * B \tag{4}$$

$$h_i = rad(a_i) \tag{5}$$

$$y_k = \frac{\sum_{q=1}^{Q} h_q . w_{kq}^O}{\sum_{q=1}^{Q} h_q} \tag{6}$$

$$\boldsymbol{W}_k^O = \left[w_{k1}^0 \ w_{k2}^0 \ldots . w_{kQ}^0 \ \right] \tag{7}$$

$$\boldsymbol{W}^O = \begin{bmatrix} w_{11}^O w_{12}^O \ldots \ w_{1Q}^O \\ w_{21}^O w_{22}^O \ldots \ w_{2Q}^O \\ \ldots \ldots \ldots \ldots \ldots \ldots .. \\ w_{N1}^O w_{N2}^O \ldots \ w_{NQ}^O \end{bmatrix} \tag{8}$$

The training process is done simply in two steps. First, define the GW, which is the width of the Gaussian transfer function. Second, set the values of $\boldsymbol{W}^I$ and $\boldsymbol{W}^O$ to be the inputs and outputs, respectively, of the training cases. Each training case consists of a set of input x and output y. For the nth training case, the nth row of $\boldsymbol{W}^I$ takes the input vector x, while the nth column of $\boldsymbol{W}^O$ takes the output vector y. The remaining question in constructing and training the GRNN is how to define the GW for such a network.

3 Method

The network parameter (GW) is heuristically and automatically determined at any application. GW has a significant effect on the predicted results of the GRNN. The mathematical importance of the GW is that when the distance A_i equals GW, then the radial function's output (h_i in Equation 5) equals 0.5. A larger GW results in a wider radial function and vice versa. Given the application-dependent nature of ANNs, it is impossible to calculate the optimal GW [10]. Having large GW decreases the accuracy of the output. While small GW provides accurate results for the training grid points, it provides poor results when evaluating off-grid points. Thus, selecting the most appropriate GW is a tradeoff between covering all space between training cases and getting accurate results. GW also determines the bias (B) in the first hidden layer, shown in Equation 9. The factor of 0.833 is provided in the literature [8].

$$B = 0.833/GW \tag{9}$$

The heuristic strategy for finding the best GW is incorporated in the training and construction of the network. The collected training cases are split into two parts, testing cases and true training cases, where the true training cases have all training cases except three cases (testing cases) that are used to test the network performance and never participate in the training. Then, 40 GRNNs are created or built, one for each of 40 different GWs. The GW values range from 0.05 to 2 in increments of 0.05, where this range represents all possible GW values within the inputs range. This range of GW values is chosen because the inputs are all normalized between -1 and 1, so the maximum difference is 2. The selected increment is small enough to exactly follow and specify the most accurate GW. Larger increments might pass the proper GW, while lower increments are useless because they are too small to have a notable effect on the produced network. The GW should also be positive. Next, for each network, R-square values are calculated between the predicted results from the network and the exact postures for the three selected testing cases and three other randomly selected on-grid (training) cases. Then, average R-square value is calculated for all six cases. The final step is choosing the maximum value in the vector, which corresponds to the best GW. By the end of this step, the training and testing steps are finished, and the best GW value for proper network performance has been identified.

Now, the proposed network along with the approach of finding its GW is applied on human posture prediction. The task is to touch a target point in front of the body with the right hand. The GRNN is used to predict 41 DOFs, which represent the upper body of the 55-DOF human model (Santos), to reach the fed target position. Figure 2 shows the 31 collected true training cases, which are randomly collected from the whole reachable zone on the front side of the body. For this task, the network has 3 inputs (target position in three dimensions) and 41 outputs (41 DOF). The optimal GW equals 0.25. The work of collecting the training cases and training the network was done on a Windows 7 computer with an Intel® Core™ 2 processor and 8 GB of RAM; the training and testing was done in a fraction of a second.

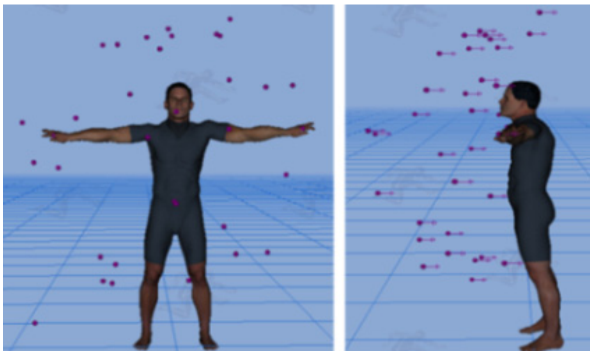

Fig. 2. Santos with the points that are used in the training cases shown in red; the figure has a front view (shown on left) and a side view (shown on right)

4 Results

Three off-grid cases are tested. On-grid points are not presented because testing the prediction of off-grid points reflects the general network prediction ability for any point in the training grid. Thus, if the GRNN predicts the off-grid points well, the general prediction ability for the network will be good. In addition, the network predicts on-grid points more accurately than off-grid points. The testing cases are selected randomly within the limits of training space (shown in Figure 2) and evaluated visually as well as statistically. Regarding the statistical results, adjusted R-squares are plotted for the three off-grid cases in Figure 3. These plots are for joint angles resulting from the network and actual Santos posture prediction outputs. The R-square values for all cases are above 0.97, which is highly acceptable from a statistical standpoint. Although these cases were not trained on the network, the results showed matching between predicted and actual values. These plots indicated that the network was able to interpolate all body joint angles properly to get acceptable accuracy. These results are statistically promising; the network is able to predict all joint angles quickly and accurately.

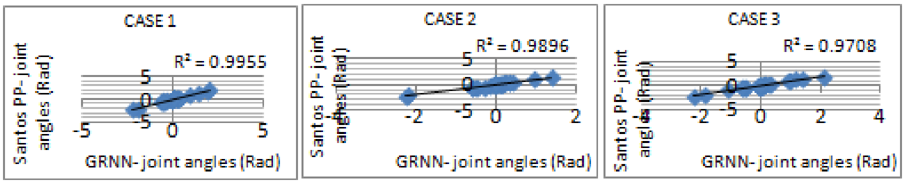

Fig. 3. Adjusted R-square values for the results of GRNN versus Santos PP for 41 DOFs of three off-grid testing cases

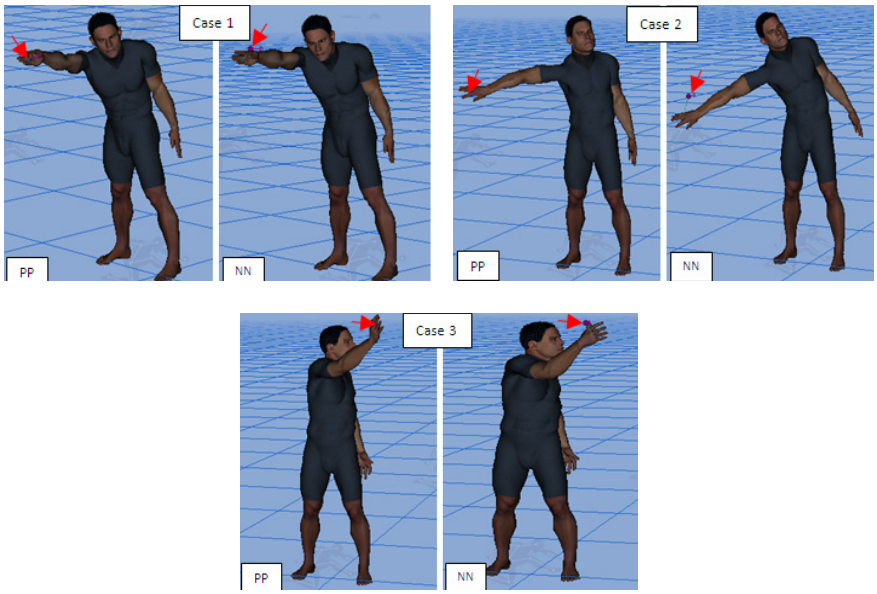

Fig. 4. Three off-grid postures in the task of touching a point for Santos posture prediction (PP) and GRNN

Visual comparisons for the three off-grid postures between posture prediction (exact posture) and predicted GRNN results are shown in Figure 4. In the figure, the red arrows refer to the target point locations. Note that there are some small errors in GRNN prediction for the joint angles, which lead the hand away from the required exact point location. The error was because the network is a general regression type, which interpolates between training cases, and it was not trained to predict the input target point with 100% accuracy. In all cases, Santos's hand moves toward the proper direction and close to the target point but with a small error. In general, the results show that the predicted joint angles, including the head and neck, were all tuned with the body corresponding to the target point position. The error in predicting the joint angles was minor, but still has a clear effect on touching the target point.

5 Discussion

A study of posture prediction for touching a point using GRNN showed that there is a potential use of the GRNN type of ANN for quick prediction of realistic postures. This prediction could be also generalized in a task-based manner and by training the network using various sources like motion capture. The results from this study were statistically promising since the network was able to predict 41 DOFs quickly and accurately, but with small error that prevents touching the target points exactly. Note that these small errors in GRNN prediction for the joint angles can manifest themselves as significant errors in the space (x, y, and z space). In all cases, Santos's hand moves toward the proper direction and close to the target point but with a small error.

The contact problem (touching an exact point or location) in this study was serious. Santos failed in exactly touching the target point for most of the tested cases. This problem occurred because the network GW value was relatively large for a task requiring highly accurate prediction ability from the network, touching a point exactly in the space. However, the GW value determined by the new heuristic method was reasonable for this task, because there were only 31 points collected for training from the whole reachable zone in front of the body, which left many gaps (empty spaces) between the training cases. Consequently, the used GW must be large enough to successfully predict the points that are located in these gaps. The large GW used in this study decreased the accuracy of predicting both on- and off-grid points. Generally, to solve the contact problem in posture-prediction tasks, there are two options: 1) collecting many training cases to decrease the gaps in the training grid or 2) adding constraints to the network construction to force the predicted postures from the network to be exactly in the proper position.

Along with the promising use of GRNN in posture prediction, there are some challenges and limitations in its current use that need to be addressed in future work. First, the accuracy of touching the target point was a problem when using GRNN. The type of inputs and outputs that form a task could be studied to improve the performance of the network used to maximize the accuracy. For example, training the network to predict joint center locations instead of joint angles has prospective success in producing more accurate results in posture prediction tasks. Second, the proper number of training cases to be chosen for such a task needs to be addressed.

Acknowledgement. This work is supported by the Office of Naval Research. The authors would also like to acknowledge the teamwork of the Virtual Soldier Research group at The University of Iowa.

References

1. Collobert, R., Weston, J.: A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In: Proceedings of the 25th International Conference on Machine Learning, pp. 160–167. ACM (2008)
2. Zha, X.: Soft Computing Framework for Intelligent Human-Machine System Design, Simulation and Optimization. Soft Computing 7(3), 184–198 (2003)
3. Zhang, X., Chaffin, D.: A Three-Dimensional Dynamic Posture Prediction Model for Simulating In-Vehicle Seated Reaching Movements: Development and Validation. Ergonomics 43(9), 1314–1330 (2000)
4. Abdel-Malek, K., Yu, W., Jaber, M.: Realistic Posture Prediction. SAE Digital Human Modeling and Simulation (2001)
5. Rezzoug, N., Gorce, P.: Prediction of Fingers Posture Using Artificial Neural Networks. Journal of Biomechanics 41(12), 2743–2749 (2008)
6. Zhang, B., Horváth, I., Molenbroek, J.F.M., Snijders, C.: Using Artificial Neural Networks forHuman Body Posture Prediction. International Journal of Industrial Ergonomics 40(4), 414–424 (2010)

7. Abdel-Malek, K., Yang, J., Kim, J., Marler, T., Beck, S., Swan, C.: Development of the Virtual Human Santos^{TM}. Digital Human Modeling 4561, 490–499 (2007)
8. Wasserman, P.D.: Advanced Methods in Neural Computing. Van Nostrand Reinhold, New York (1993)
9. Buhmann, M.D.: Radial Basis Functions: Theory and Implementations. Cambridge University Press, Cambridge (2003)
10. Specht, D.F.: A General Regression Neural Network. IEEE Trans. Neural Network 2(6), 568–576 (1991)