NEURAL NETWORK MODELING IN SPORT BIOMECHANICS BASED ON THE EXAMPLE OF SHOT-PUT FLIGHT

Klaus D. Maier, Petra Meier, Heiko Wagner and Reinhard Blickhan
Friedrich-Schiller-University Jena, Germany

In this study, a modeling method based on Multi-Layer-Perceptron neural networks (MLP) is presented, using the example of shot-put flight. This method can be used for rapid construction of models. For performance of shot-put, a physical model based on the shots' equations of motion is easily designed. In this way, an analysis of the shot-put projectory can be used to illustrate the effectiveness of the neural network modeling method. Using the physical model, release data has been determined and altered with random errors such as those introduced by video analysis. A technique for optimal learning with the neural networks has been developed. The resulting MLP models the shot-put flight successfully. The difference between the model's predicted distance and the distance reached by the physical model are within 2.5%. In conclusion, this method allows rapid creation of models to solve biomechanical problems and can serve as a useful tool for coaches and athletes.

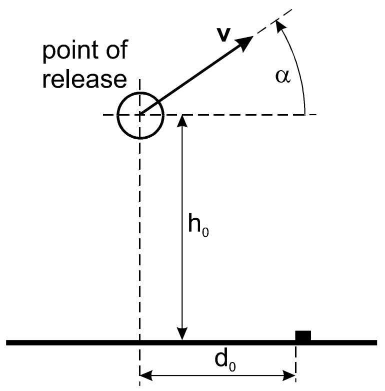
KEY WORDS: modeling, neural networks, shot put

INTRODUCTION: The design of models for biomechanical problems can be an extremely laborious and sometimes daunting task. This can be due to imprecise data acquisition for the model, inherent non-linearities and the overall complexity of the task. It can also be caused by influences from the environment that can not be determined at all, or only measured with extreme difficulty. However, in sports biomechanics there is a need for models to support coaches and athletes with the aim to improve performance. This paper presents a modeling method that can be used to construct rapidly, an input-output model of a problem by utilizing the example provided by input-output data. This method is based on the powerful information processing technology subsumed under the term artificial neural networks (NNs; Haykin, 1994; Jain & Mao, 1996). NNs are able to generalize, i.e. to produce reasonable outputs for inputs that have not been encountered during learning. They also handle fuzzy, faulty and imprecise data adequately. This method is presented by using the example of shot-put flight. The objective of throwing discipline models is to find the optimal release parameters for achieving a maximal flight distance. A physical model of shot-put flight is designed easily based on the shots' equation of motion. Analysis of actual shot-put action shows that there are only very small differences between the result of the physical model and the measured data. Therefore, choosing the shot-put task allows verification of the neural network model by using the data from the physical model. For more complex tasks, the verification would be limited to data measured or generated from other models. In a study by Maier et al. it has been shown that artificial neural networks, successfully can be used to model a complex problem like javelin flight, with strong non-linearities due to the javelin design and wind influence (Maier et al.). It has been problematic, verifying the performance for input data ranges where no measured data was available. Using existing models that are designed with classic methods is also problematic as these too are based on imprecise measurement data and so differ significantly from reality in some areas (Hubbard, 1984).

METHODS: For throwing disciplines, release parameters are normally measured by using video analysis. For analysis of shot put flight, relevant parameters are the angle of release α and the velocity of release $\mathbf{v}$ (Figure 1a). The overall flight distance d is measured as the distance between the throwing line and the athlete's hand at the point of release, d_0 plus the distance between the line and the point of touch down of the shot. Data patterns with the resulting distances were generated using the physical model for release angles $30° \leq \alpha \leq 55°$. These were recorded in steps of 1° and the release velocities $9\text{m/s} \leq v \leq 15\text{m/s}$ were in steps of 1m/s. From these, 100 shot-put attempts were selected at random to serve as 'measured' data for creating the NN model. Typically, the precision error encountered using video analysis is for angular data approximately 1° and approximately 0.1 m/s for velocity data. The release parameters for the selected puts had a random error in the dimension of the precision error which was added to generate realistic measurement data. The physical model used a fixed release height h_0 of 2m. Wind speed and aerodynamics of the shot, etc., were not considered in the model.

Multi-Layer-Perceptron neural networks (MLPs; Jain & Mao, 1996) are used to construct a model with the release parameters as inputs and the overall distance as output (Figure 1b). MLPs are made up of 'neurons' that have a number of inputs and generate an output using a non-linearity. Neurons in a MLP can be divided in input, output neurons and neurons that are neither one nor the other, so called hidden neurons. The whole structure is grouped in layers of neurons, i.e. input layer, output layer and a number of hidden layers of neurons that can be seen as parallel processing units. The hidden layers enable the MLP to learn complex tasks by extracting properties from the input-output patterns. One neuron in a layer is connected with all neurons in the following layer using connections with weights. The input of a neuron consists of the outputs of the previous layer, multiplied with each of the connection weights. During 'training', the weight of the connections is adjusted by means of the Levenberg-Marquardt algorithm (LM; Hagan & Menhaj, 1994). LM can be seen as a gradient-descent method, required to find a good weight configuration by minimizing an error cost function, with repeatedly presenting input-output patterns starting from randomly distributed weights. One presentation of the pattern set is called an epoch. Once a MLP is trained, it has input data fed into it, and from that, generates an appropriate output (retrieval phase). The structure of the MLP, combined with the non-linearity and its weights, adjusted by the training algorithm, creates general function approximations that allow for generation of practically every non-linear function.

An important issue with this method is the ability to generalize: MLPs can be 'over-trained' easily using LM. The result of this is closely matching outputs for trained data patterns but in poor outcome for untrained patterns. To avoid this problem, a form of cross-validation technique (CV; Stone, 1974), can be used, by choosing good weight configurations during learning. The original set of training data is split into three groups: The first group consists of training a data set (TDS) with about 75% of the data patterns. The remaining groups are the two CV-sets: the validation data set (VDS) with about 20% of the patterns and the remaining approximately 5% of the data patterns, to evaluate the performance (EPS) of the MLP. With the CV technique, the data patterns of TDS are used for training the MLP with LM. After each training epoch the MLP's performance is tested using the VDS. The weight configuration for the best performance reached is stored and only replaced if a better performance has occurred. In this way, the best generalizing weight configuration can be determined. To reduce the influence of the randomly selected starting weight configuration, this learning technique is repeated a number of times. Finally EPS can be used to evaluate the performance of each MLP used and to select the best operating MLP setup of the ones investigated.



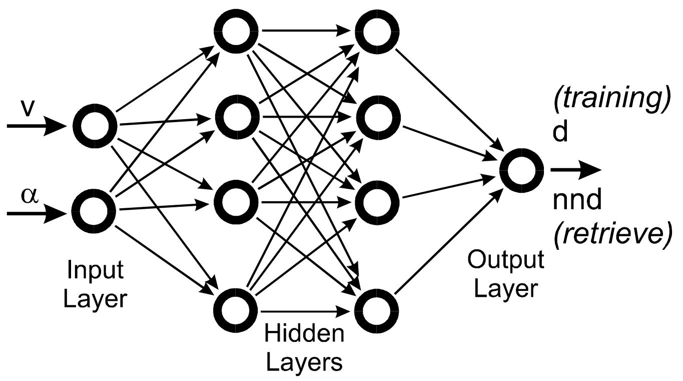


Figure 1a - Release parameters; 1b - Structure of the MLP. Inputs are the release parameters, outputs are the overall distances d when training and the simulated distance nnd when retrieving.

RESULTS: This study experimented with several MLP setups using two hidden layers with approximately two and 10 neurons in each layer. A small setup that showed good results consisted of two input neurons (matching the number of input data), one output neuron (the total flight distance), five neurons in the first and 3 neurons in the second hidden layer. The learning algorithm was finally stopped when the summed square error for all TDS dropped below 0.1m. The VDS was used to test the generalization performance of the MLP – the weight configuration of each MLP setup was saved when the mean differences between the predicted distance nnd and the measured distance d was at a minimum. This was repeated 200 times for each setup in order to reduce the influence of the randomly selected starting weights for the minimal search. The final prediction quality was tested on the EPS. The MLP setup best suited for the modeling task is the one with the lowest mean error between the predicted distance ndd and 'measured' distance d. The whole experiment was repeated four times with different training sets to ensure reproducibility and all four experiments yielded similar results in all repetitions.

Figure 2 shows the distances resulting from the above MLP as function of the release velocity and the release angles for the above MLP.

Figure 3 shows the differences between the simulated distances and the distances determined using the physical model in absolute percentage. The simulated flight distances ndd are accurate up to about 2.5 percent.

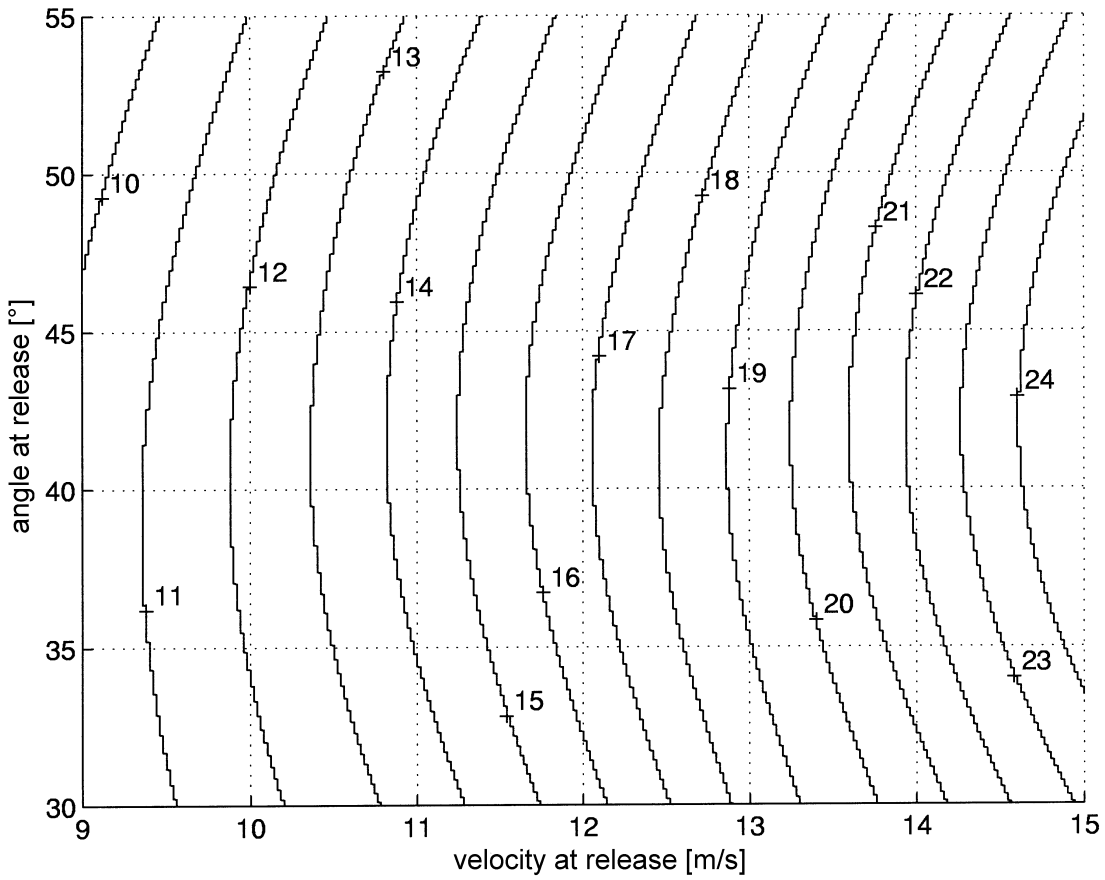


Figure 2 - Resulting simulated distances ndd shown as function of release velocity and release angle.

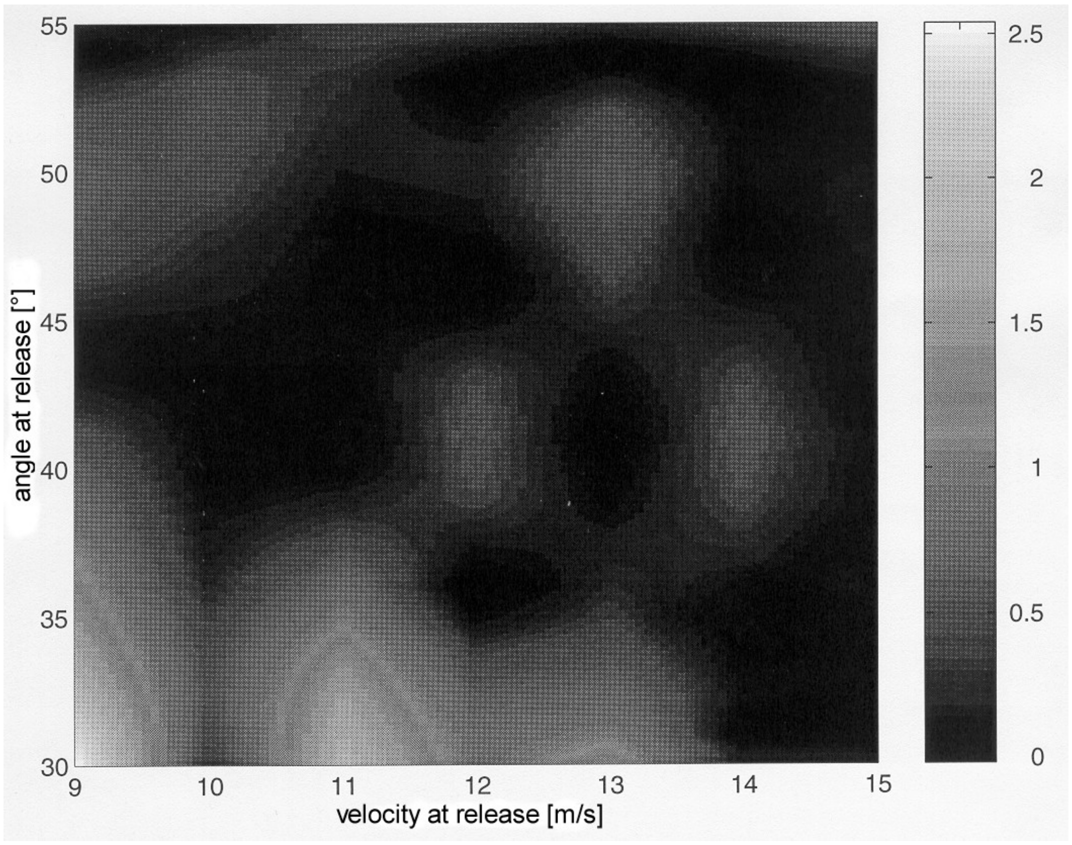


Figure 3 - Differences between predicted distances and distances determined by the physical model in percent.

CONCLUSION: It was demonstrated by this study, that the neural network based approach presented is a suitable instrument for the design of models. In the case of the modeled shot-put flight, the results are accurate within 2.5% over the whole covered range. The method that has been presented here can be used to generate other models for complex and non-linear problems. It allows implicit physical modeling without describing the explicit physical principle. Therefore, it can help to create models rapidly and provides a practical alternative to classic bio-mechanic modeling techniques. At times, these have been considered to be too demanding with regards to time involved, affected by a shortage of resources or techniques that may not be considered viable at all. Once such a model has been created it can also be used in training, to assist coaching.

REFERENCES:

Hagan, M., & Menhaj, M. (1994) Training feed-forward networks with the Marquardt algorithm, *IEEE Transactions on Neural Networks*, **6**, 989-993.

Haykin, S. (1994). *Neural Networks – A Comprehensive Foundation*, Englewood Cliffs, Macmillan Publishing Company.

Hubbard, M. (1984). Optimal Javelin Trajectories. *Journal of Biomechanics*, **17**, 777-787.

Jain, A.K., & Mao, J. (1996) Artificial Neural Networks: A Tutorial. *Computer*, **3**, 31-45.

Maier, K.D., Wank, V, Bartonietz, K., & Blickhan R. (in print) Neural Network based Models of Javelin Flight: Prediction of Flight Distances and optimal Release Parameters, *Journal of Sports Engineering*.

Stone, M. (1974) Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society*, **B36**, 111-133.