Behavioral Cloning of Student Pilots with Modular Neural Networks

Charles W. Anderson ANDERSON@CS.COLOSTATE.EDU
Department of Computer Science, Colorado State University, Fort Collins, CO 80523 USA

Bruce A. Draper DRAPER@CS.COLOSTATE.EDU
Department of Computer Science, Colorado State University, Fort Collins, CO 80523 USA

David A. Peterson PETERSOD@CS.COLOSTATE.EDU
Department of Computer Science, Colorado State University, Fort Collins, CO 80523 USA

Abstract

This paper investigates how behavioral cloning can be used to decrease training time for students learning to fly on simulators. The challenges presented to each student must be tailored to their unique learning experiences. This requires an intelligent training regime that exploits a model of each student that predicts where the student's performance will be deficient. Here we show that cloning the behavior of student pilots with a modular neural network results in the automatic decomposition of the behavior into sets of skills. This decomposition may provide a means for identifying when certain skills are acquired by students and which skills are deficient. This information may then be used to decrease training time by altering the sequence of simulation experiences to just those that the student needs.

1. Problem

In 1995, the FAA and NASA formed a consortium to advance the concept of a small aircraft transportation system. The consortium's primary focus is the Advanced General Aviation Transport Experiments (AGATE). The goal of this government-industry-university partnership is to develop technologies for safety, affordability, and ease of use for single pilot, single engine, near all weather transportation aircraft and related training, airspace, and ground infrastructure systems. Targeted training technologies include cockpit-based onboard training systems, computer-based training desktop devices, simulation, virtual environment systems, and expert systems. Teaching a person to fly a general aviation airplane is a complex and time consuming task. The approach of using one-on-one training with an instructor in the co-pilot's seat of a real airplane has been used for decades. While real aircraft offer the ultimate in training realism, they are expensive to operate and maintain, and safety considerations restrict the tasks that can be practiced. Augmenting instructor training in a real airplane with computer simulation promises to reduce training costs and allow instruction in emergencies in a forgiving environment.

We investigated augmenting simulation capabilities with adaptive techniques that model the abilities of each student pilot. Ultimately the ability to predict each student's capabilities and to adapt the simulation according to their strengths and weaknesses will reduce the number of human instructor hours required and allow for better training over a shorter period of time. Our first thrust at this broad problem was to develop a system that builds dynamic models of a student's current flying ability and predicts their trajectory in simulations.

2. Objective

If large numbers of new pilots are to be trained for the general aviation market as envisioned by the NASA/FAA AGATE project, then the training process for new pilots must be made cost-effective. This implies that as much of a pilot's training as possible should take place on low-cost simulators, thereby reducing the need for expensive in-flight supervised instruction. Our long-term goal of an intelligent training system for general aviation is to maximize the training benefits of simulations.

The immediate goal of this project was to lay the foundation for a future intelligent training system by developing techniques to model the skills and performance

of individual students. These models can be used to develop intelligent training systems that 1) select training scenarios that challenge but not overwhelm the student, 2) selectively exercise skills in which a student is deficient, while only occasionally reinforcing previously mastered material, and 3) actually exert forces on control surfaces in situations where the student consistently makes mistakes, like a golf instructor helping a novice player with their swing. Such intelligent training systems, however, will rely on techniques such as those developed in this project to 1) develop accurate models of individual pilots and 2) verify that the learning process of pilots differ sufficiently to warrant individualized training regimens.

This project developed techniques for modeling the performance of individual novice pilots. Whereas most intelligent tutoring systems model students as decision trees or through sets of production rules, we believe these techniques may not be effective in the real-time, continuous-process world of general aviation. We therefore developed control-model techniques which learned a mapping from the current state of the plane (including its recent history) to the pilot's predicted action. Note that we are learning to predict what the pilot *will* do, not what they *should* do. The former is learning a model of the pilot, whereas the second would be learning to fly the plane.

One challenge in developing these control models is to select a state space, since both the plane's current position/orientation/speed and the recent history of these variables may determine a pilot's action, particularly given varying reaction times and the tendency of many novice pilots to over-compensate for small errors. Another challenge is to learn a non-stationary process model for the student, whose performance improves with each trial. A significant result of this work is that we verified that novice pilots vary significantly from each other in what and how they learn.

3. Approach

We built a system that develops predictive control models of individual student pilots, and compared these models across students to measure how similar or dissimilar they are at different points during a training regimen. We believe that we succeeded in modeling a pilot's behavior largely because of the "highway in the sky" concept underlying the AGATE project. Part of the motivation for AGATE is that general aviation pilots will enter a flight plan into the plane's computer before take off. The computer system will then use this information to project a virtual 3D highway onto the plane's windshield, complete with road signs for auxiliary information. In this way, flying a plane becomes much more like driving a car; pilots simply keep the plane slightly above the (virtual) highway, and slow down or speed up when (virtual) road signs tell them to. A mock-up of the "highway in the sky" is shown in Figure 1.

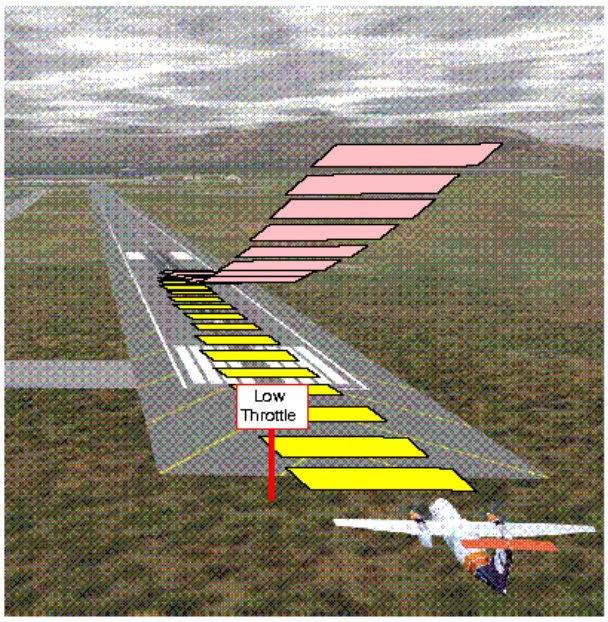

Figure 1. Highway in the sky.

The virtual skyway reduces flying to a reactive task. Pilots react to the projected highway in front of them, and the act of flying becomes the hand-eye coordination task of keeping the plane over the skyway. This does not imply that flying becomes trivial; pilots must still control a plane in 3D to keep it on course relative to the pre-planned highway, a difficult task in some weather conditions or if the flight plan calls for relatively tight curves (for example, on take-off or landing). Nonetheless, it does make flying a primarily reactive skill in which pilots respond to the information immediately before them, rather than an abstract navigation task in which pilots respond to mental estimates of their current and desired position. Therefore, we developed a gated network of neural network "expert" controllers, each automatically tuned to a different aspect of flying, that models a pilot's behavior in terms of a small set of immediately available variables, summarizing the position, velocity and acceleration of the plane and the local position and curvature of the path.

3.1 Related Work

It should be noted that researchers have sought to develop intelligent tutoring systems since at least the early 1980's (Sleeman & Brown, 1982; Yazdani, 1986) with a great deal of recent work focusing on medical training applications (Woolf, 1996). What separates this project from most of the literature in intelligent tutoring is that we are not trying to model students as they learn abstract concepts. Instead, we are trying to teach a hand-eye coordination task, almost at the level of a motor skill. As a result, we do not need to model a student's concepts and ideas about flying, but rather their reaction time, their tendency to over-compensate of under-compensate for positional errors, and how well they control the plane. This is why we use the term "intelligent training" as opposed to "intelligent tutoring", in keeping with the terminology of the NASA Conference on Intelligent Computer-Aided Training held in 1991 at the Johnson Space Center, Houston, TX.

Most intelligent tutoring systems have modeled their students using decision trees or production rules. We believe that the domain shift from abstract ideas to motor skills would be better matched by a corresponding shift from hierarchical decision tree or production rule models of student behavior to subsymbolic models. This is why we created a gated network of neural network "experts", described below, to model the performance of novice pilots.

Much more closely related to our work is the recent work in behavioral cloning. Sammut, et al. (1992), like us, have developed models of pilot behavior. Their approach is based on decision trees, though we believe the structure of the decision trees resulting from their approach might also be used to hypothesize which skill sets govern the pilots behavior, as we do here with modular neural networks. A comparative study of our approach and that of Sammut, et al., would clearly be a fruitful next step.

3.2 Modular Neural Networks

Learning a model of a complex, dynamic process is closely related to the problem of learning to control such a process. In this section, we describe the modular network approach to control as it was first defined. Then we describe how we extended the approach to the problem of modeling a student pilot.

For the control of complex, dynamic systems, it is often impossible to define a single feedback controller that works well over all possible operating conditions. Instead, multiple controllers can be defined and activated for different parts of the system's state space. One way to do this is by *gain scheduling*, where multiple controllers are defined by specifying sets of controller parameters, or gains, and assigning their use to different regions of state space (Shamma & Athans, 1990). The designer must partition the state space into regions for which the system can be approximated by linear time-invariant models. Individual controllers can then be designed for each region.

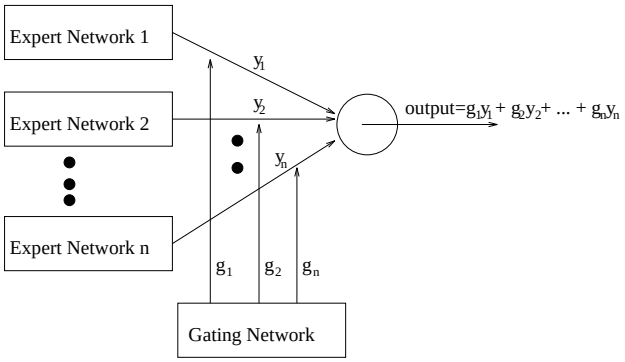

Figure 2. A modular neural network composed of one gating network and n expert networks. The expert networks learn unique, simple models. The gating network mixes the predictions of each expert to obtain an overall prediction.

Partitioning the state space appropriately requires knowledge of the dynamics of the system to be controlled. The design effort would be drastically reduced if the regions and controllers could be learned from samples of interactions between the controller and the system. Jacobs and Jordan (1991) describe such an approach. Their method consists of multiple neural networks, called *expert networks*, each learning different, relatively simple, control laws. An additional network learns when to activate the expert networks, possibly mixing, or *gating*, the responses of the individual networks to obtain an overall response from the collection of networks. Figure 2 show the configuration of a modular neural network consisting of n expert networks and the gating network.

When complex neural networks are trained to model a set of data, they are very likely to *over-fit* the data, much as a high-order polynomial will over-fit a small number of data points. The modular network approach is one way to limit the amount of over-fitting. Each expert network is restricted to the learning of a model of low complexity. The complexity of each network is limited by the number of computational units contained in the network, their interconnectivity, and their transfer functions. The gating network combines the output of the experts, but again the combination is very simple, usually just a weighted, linear combi-

nation of the outputs of the expert networks. The expert and gating networks and the combination function are parameterized by numerical weights, which can be adjusted by any optimization technique. The most common is an extension of the error backpropagation algorithm, which is simply a gradient descent in the overall network's error function.

It is rather surprising that all of the weights in the modular network can be adjusted simultaneously. In fact, by learning all the expert networks and the gating network simultaneously, the problem is automatically decomposed into pieces whose complexity is limited by the complexity of the expert networks. Jacobs and Jordan (1991) show that for a problem involving the control of a robot arm whose gripper can hold a variety of payloads, the expert networks learn simple control laws that pertain to particular payload masses. The appropriate expert network is selected by the gating network as a function of the payload identity, which does not indicate the actual mass of the payload. After their modular network was trained for a number of different payloads, they tested the network's ability to control a novel payload. They found that the gating network used a weighted combination of previously-learned expert networks to successfully control the new payload.

Another example is the work of Anderson and Hong (1994), who trained a reinforcement-learning variant of the modular network architecture to balance a simulated inverted pendulum. Analysis of the resulting modular network showed that the expert networks did automatically decompose the task. In most runs, the expert networks became tuned to either positive or negative velocity of the cart. Other decompositions were also found.

For our project, we learned models of student pilots using a modular neural network. In this case, the modular network learned to predict the actions of the students, given as input the current variables that the students can observe. The ability to automatically decompose a problem is very useful in this context. In addition to controlling for over-fitting, the resulting decomposition forms a categorization of student behaviors. For example, one expert network might learn to predict student actions on take-offs, while another might learn to predict actions for level flight. This result would be very useful to SymSystems for future intelligent training systems. Fundamental behaviors that are identified would form the basis of the adaptive formation of training regimes for each student.

We expect particular behaviors to be modeled by the expert networks if the networks' complexity is maintained at a fairly low level. If we allow the expert networks to be somewhat more complex, simple behaviors would no longer be learned; instead, whole repertoires of behaviors would be modeled by each expert network. Each expert network would become specialized to the nuances in behavior exhibited by classes of students. Perhaps one expert network would become tuned to predicting the actions of skilled pilots, while another would predict actions of novices. If certain students have more trouble with left and right control while others have trouble with up and down control, then we would expect separate expert networks to become tuned to these two classes of students.

Our study of modular networks was constrained to simple feedforward neural networks for the expert and gating networks. One hidden layer was used in each. The number of hidden units in each network can be varied to study the effects of varying the complexity of the networks. The number of expert networks can also be varied.

3.3 Data

A majority of the effort on this project was consumed by the recording of flight data and experimentation with ways of representing the data as input to the neural networks. This section of the report summarizes the data that we collected, the representation problems we faced, and our solutions to these problems.

The data recorded from the flight simulator that we used in our study is summarized in the following list:

position$_{xyz}$— position of the plane in a left-handed coordinate system referenced to ground;

paver positions$_{xyz}$— measured in the same coordinate system as the position of the plane;

velocity$_{xyz}$**, acceleration**$_{xyz}$— based on a left-handed coordinate system referenced to the plane: positive x is out the nose of the plane, positive y is out the right wing of the plane, and positive z is out the top of the plane;

pitch, roll, yaw— measured in degrees, but presented to the modular neural network as pairs of sine and cosine values to allow smooth generalization through the angular range;

pitch, roll, yaw velocities— measured in degrees per second;

SLPC— Single Lever Power Control: value ranges correspond to types of flight and to the SLPC label on the head-up display. For example, for

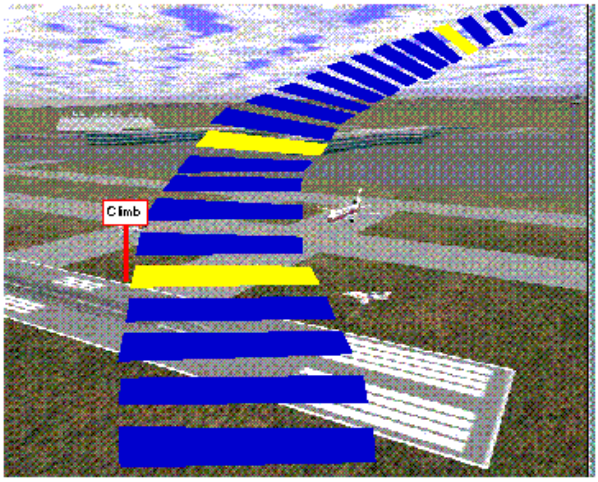

Figure 3. The 5th, 10th, and 20th pavers are shaded light gray. Their locations relative to the plane are a simple, albeit limited, way to represent the pilot's view.

"climb" the range is 207–215, for "cruise" it is 167–179, and for "descent" it is 11–29. Hereafter, the SLPC is called the *throttle*;

stick$_x$— negative is left, positive is right;

stick$_y$— negative is a pull back, positive is a push forward;

elevator tab— negative is down, positive is up.

To model a student pilot, a modular neural network must be provided with inputs that are similar to what the pilot experiences as they make their decisions about how to move the stick, throttle, and elevator. Pilots look ahead to the pavers in the "highway" to decide which direction and how quickly they must turn the plane. There are many ways to model this perception; we chose a simple method using the locations of the 5th, 10th, and 20th next pavers relative to the plane. A diagram of these pavers is shown in Figure 3. These paver positions, along with the plane's velocity, acceleration, orientation, and rotational velocity, are provided as input to the network. The output of the network is the prediction of the pilot's settings of the throttle, stick$_x$, and stick$_y$. We chose not to predict the elevator tab, because it was not widely used by the subjects.

How many steps ahead should we train the network to predict these actions? To explore this question, we trained networks to predict actions at a number of time steps ahead. There is a large increase in error as the number of steps ahead is increased from 1 to 10, beyond which there is much less of an increase.

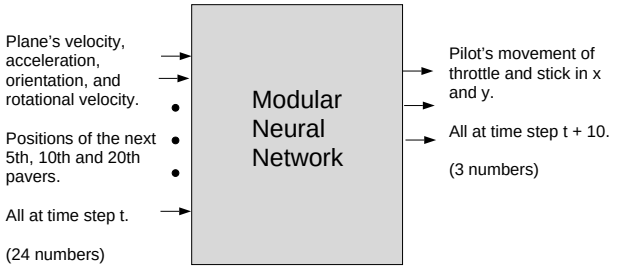

Figure 4. Modular neural network inputs and outputs. All expert networks and the gating network receive all inputs.

The sampling rate was approximately 7 samples per second. We reasoned that pilots react within one to two seconds, so we decided to predict 10 steps ahead. Figure 4 summarizes the inputs and outputs of the neural network.

3.4 Training the Modular Neural Network

Data were gathered from two subjects, A and B, flying the simulator six different times. Each time consisted of taking off, following a "figure-eight" path, and landing. Neither subject had any experience with flying a real airplane, or with flying this simulator, although Subject B had some experience with flying other simulators. The subjects showed definite improvement in ability to fly the simulator and stay on the flight path. Improvement was not consistent from one flight to the next, but each subject had clearly learned a great deal about how the simulated aircraft handled by the sixth flight.

We trained a modular neural network on data from all twelve flights. The modular network had six expert nets, each having two hidden units, and the gating network had five hidden units. Input components were normalized to have zero mean and unit variance. Output components were scaled to be between 0.1 and 0.9. All units in the network used logistic sigmoid activation functions. Error backpropagation was used to train the modular network for 200 epochs. Training was performed using 80% of the data (30,832 samples) and 10% (3,854 samples) was used to evaluate the training progress after every epoch. The best epoch was the last, Epoch 200. The final 10% (3,854 samples) of the data comprised our test set. The RMS error on this test set was 0.056, which is about 6% of the allowed output range of 0.1 to 0.9. The learning rate for hidden units was 0.1, and for output units, 0.01. A momentum factor of 0.9 was used. Initial weights ranged between -0.1 and 0.1.

4. Results

The first three graphs in Figures 5 and 6 show the actual and predicted actions for Subject A's Flights 3 and 6. Note that the general trends of the actions are fairly well predicted, but the large, fast changes in stick x and y are not matched very well. This may be due either to a deficiency in the network's input or in the network itself. The input is deficient if the input variables do not adequately represent all of the information known by the pilot; our prediction of actions may not be based on the whole context available to the pilot. The neural network would be deficient if too few expert networks are used, or if each expert network contains too few hidden units.

The fourth graphs in Figures 5 and 6 plot the index of the gating network which has the strongest influence on the predicted output for each sample in time. This

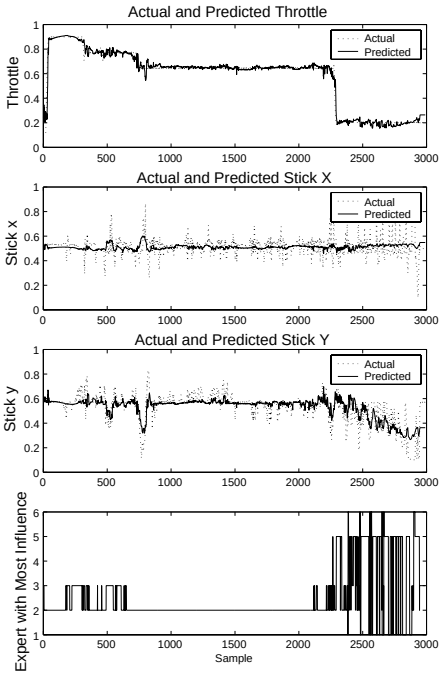

Figure 5. Subject A's Flight 3. Actual and predicted control actions are plotted in the first three graphs and the expert network most responsible for the predictions is plotted for each sample in the fourth graph.

index is simply the index of the gating network output with the largest magnitude, and indicates which expert network has the most responsibility in forming the prediction for each sample. During the middle part of each flight when the pilot is primarily cruising level, Expert Network 2 is mostly responsible. During takeoff and landing, other expert nets come into play. By Flight 6, this pilot has developed enough skill to avoid

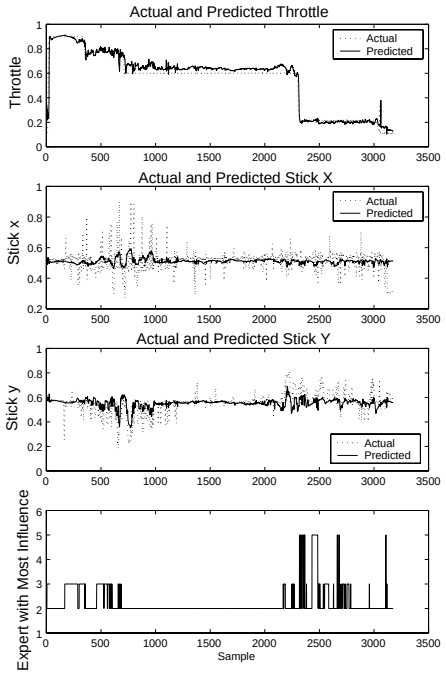

Figure 6. Subject A's Flight 6. At this stage, Subject A is learned better control of the airplane. Fewer expert networks are needed to predict the behavior.

some of the erratic behavior seen in Flight 3 and modeled by other expert networks. Flights for Subject B are similar.

It is difficult to judge from these graphs which expert net is best at predicting actions for any sample. A much easier way is to draw a 3-dimensional view of the plane's position and orientation at various samples and indicate for each sample which expert network is best at predicting the actions. This view is shown for one of Subject A's flights in Figure 7. Uncircled samples are predicted best by Expert Network 2. The second and third most influential expert nets are Expert Nets 3 and 5. Samples for which each is most influential are marked by circles and ovals and are usually situations where the pilot must dive, climb, or swerve left or right to get back to the path marked by the pavers. With practice, the pilot learns to deviate less from the flight path, resulting in more samples that are well-modeled by the primary network, Expert Network 2, and fewer samples modeled by Expert Networks 3 and 5.

The hypothesis driving this study is that patterns will be present in the distribution of expert network responsibilities as a pilot learns to fly the simulator and that these patterns will suggest how to structure further training. To investigate our hypothesis, we plotted the distributions of network responsibilities for

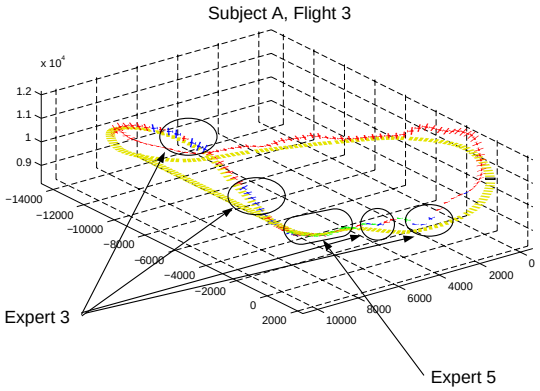

Figure 7. Subject A's Third Flight. Circles mark situations for which particular expert nets were most responsible for predicting actions. Most situations are uncircled and are predicted by the second expert network.

each of the six flights for our two subjects. Figure 8 shows the resulting histograms as the count of samples for which each expert network was most responsible. The vertical axis has been truncated to 1,000; the maximum counts are close to 3,000.

The distribution for Subject A varies considerably from the first through the sixth flights, whereas Subject B's distributions are relatively constant. This may be due to Subject B's prior experience with flight simulators. Subject A's distributions for Flights 5 and 6 are a close match to Subject B's, suggesting that this distribution pattern is indicative of a student who has progressed beyond the novice stage.

How could this information be used to tailer further training? The first four flights by Subject A generated more samples predicted by Expert Networks 1 and 5 than did later flights. This suggests that it might have been a more efficient use of time if, during Flight 2, Subject A had been primarily exposed to situations for which Expert Networks 1 and 5 were more responsible and less to situations for which Expert Network 2 is responsible. This possibility has not been tested yet.

One perceived difficulty in using neural networks to model behavior is that the interpretation of what each expert network has learned is not as straightforward as it is with more symbolic representations. A start at an interpretation can be made by studying the weights learned by the expert networks. The six expert networks used in this study after training are shown in Figure 9. The six expert networks are arranged in six columns. Within each column, the weights of the two hidden units are drawn vertically. The outputs of the two hidden units in each expert network provide inputs into the three output units whose weight vectors

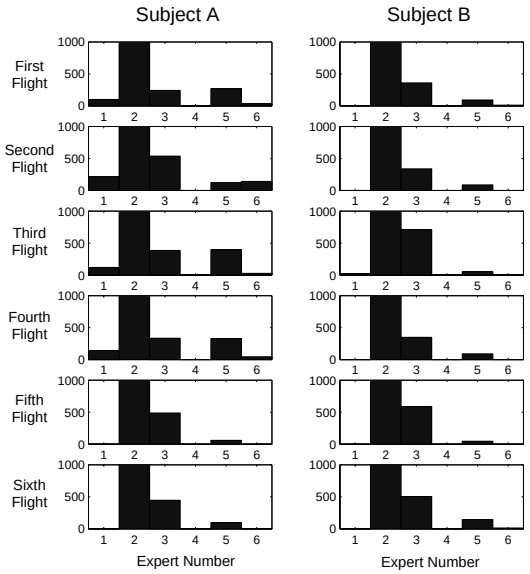

Figure 8. Histograms for flights 1 through 6 for Subject A are shown on the left. Flights 1 through 6 for Subject B are shown on the right.

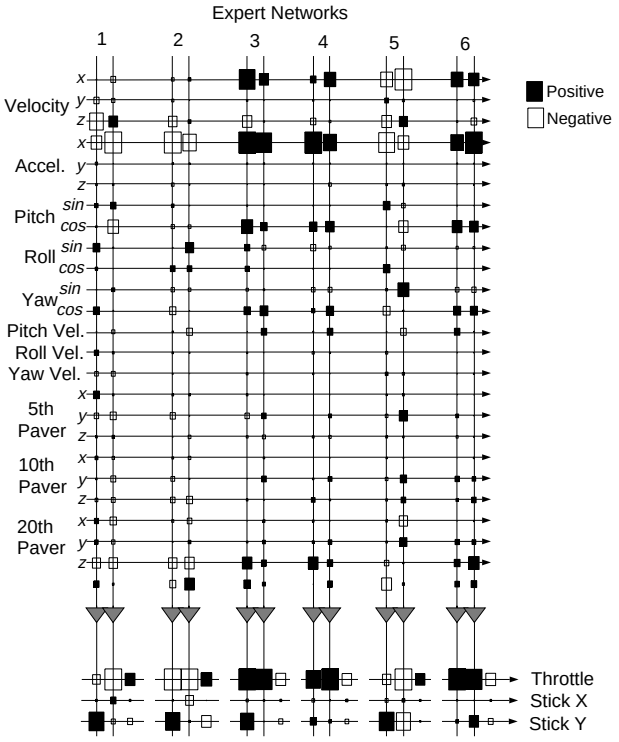

Figure 9. Weights of the expert networks after training.

are drawn in rows. Positive weights are drawn as filled rectangles and negative weights are unfilled.

Let's examine Expert Network 2. Here we find that both hidden units have large negative weights for the acceleration in the x direction and both units are connected negatively to the throttle output. This results in Expert Network 2 making the obvious prediction that when the forward movement of the airplane is slowing down, the throttle is set to a low value. Expert Networks 1 and 5 make the same throttle predictions and Expert Networks 3, 4 and 6 make similar predictions, but opposite in sign—positive forward acceleration predicts high throttle settings.

More interesting are the predictions of stick settings. The first hidden unit of Expert Network 2 is connected positively to stick_y. This hidden unit's activation is increased when the upcoming pavers are at negative z values relative to the plane, meaning that the pavers are below the plane. Therefore, this unit models the result that the pilot pushes the stick forward when the plane is too high. It is a bit more difficult to interpret the effect of the other hidden unit in Expert Network 2. It is connected negatively to stick_x and is more activated by a positive value of the sine of the roll angle. This is probably representing a correlation between rolling to the left and a left push on the stick, rather than a prediction of the pilot's decisions.

5. Conclusions

These initial results prompt more questions than answers. Data for more subjects and more flights are clearly needed in order to draw valid conclusions. The results we have obtained so far suggest that this may be a fruitful approach towards intelligent training. A system that adapts to the learner's acquired skills could greatly reduce training time by focusing on experiences that are predicted to be troublesome for the student. This has yet to be tested for this approach.

Limitations of this approach include the obvious difficulty in interpreting what has been learned by each module. We believe that our analysis of the trained weights is hindered by the non-discrete mixing of experts' outputs by the gating network. Interpretation should be considerably easier if a bias towards a more discrete selection of experts were built into the training algorithm, which might be obtained by adding a winner-take-all selection of experts networks on each step, or by adding a term to the error function being minimized that is inversely proportional to the entropy of the outputs of the gating network.

Acknowledgments

This work was supported by a grant from the Colorado Advanced Software Institute. The authors would also like to thank SymSystems, of Englewood, Colorado, (www.symsystems.com) for making their simulator available to us and for collecting the data.

References

Anderson, C. W. & Hong, Z. (1994). Reinforcement learning with modular neural networks for control. In *Proceedings of the AMCA/IEEE International Workshop on Neural Networks Applied to Control and Image Processing.*

Jacobs, R. A. & Jordan, M. I. (1991). A modular connectionist architecture for learning piecewise control strategies. In *Proceedings of the 1991 American Control Conference.*

Sammut, C., Hurst, S., Kedzier, D. & Michie, D. (1992). Learning to fly. In *Proceedings of the Ninth International Conference on Machine Learning* (pp. 385–393). Aberdeen: Morgan Kaufmann.

Shamma, J. S. & Athans, M. (1990). Analysis of gain scheduled control for nonlinear plants. *IEEE Transactions on Automatic Control*, *35*, 898–907.

Sleeman, D. & Brown, J. S. (Eds.). (1982). *Intelligent tutoring systems.* London: Academic Press.

Woolf, B. (1996). Intelligent multimedia computer tutors. *Communications of the ACM*, *34*, 30–31.

Yazdani, M. (1986). Intelligent tutoring systems survey. *Artificial Intelligence Review*, *1*, 43.