A Method of Automatic Training Sequence Generation for Recursive Neural Networks in the Area of Sensing Directional Motion

Dudley Girard
Computer Science Deptartment
University of South Carolina
girard@cs.sc.edu

Abstract
In the modeling of vision systems of biological organisms, one of the important features is the ability to sense motion (Borg-Graham et al. 1992, Huntsberger 1995, Klauber 1997, Missler and Kamangar 1995, Rosenberg and Ariel 1991). Motion is sensed by animals through neurons that receive input over some area of the field of view (Newman et al. 1982 and Rosenberg and Ariel 1991). For such a neuron to function properly implies the ability to "remember" how things were in the past and combine that information with how things are in the present. In attempting to come up with a computationally efficient model of a motion neuron a Jordan-Elman network has been utilized. The Jordan-Elman network allows for a one time step remembrance of the state space, making it suitable for motion sensing (Jordan 1986 and Elman 1990). To train the network an automatic method of building a training sequence was developed based off earlier work done mostly through hand coding to allow for easy construction of motion sensing networks of differing features. A network with a 3x3 input field and a network with a 5x5 input field were trained using the automatic method of training sequence generation. These two networks were tested against a network with a 3x3 input field trained using the training set constructed partially by hand. Results favored the latter network, but the former networks showed future promise. It is hoped with the right modifications to the creation of their training sequence they will become better than their hand built ancestor.

Introduction

Sensing motion is an important feature of a biological organism's vision system. Within the turtle it is known that there are neurons that respond to motion in any direction, neurons that respond to motion along an axis, and neurons that respond to motion only in a certain direction (Rosenberg and Ariel 1991). Neurons that respond to motion along a certain direction tend to have an area of preference. For the rattlesnake this has been shown to be +/- 20 degrees off the preferred direction and for the turtle it is +/- 60 degrees (Hartline et al. 1978 and Rosenberg and Ariel 1991). Also for motion sensing neurons the input field of view varies from neuron to neuron within the animal and between different animals (Hartline et al. 1976 and Rosenberg and Ariel 1991). These are the features that the Jordan-Elman networks discussed in this paper are attempting to mimic within a local area of the viewing space.

Training for the two newer networks was accomplished through an automatically generated nondeterministic finite automaton (NFA) whose transitions were assigned weights based on whether motion occurred or not. The underlying design for these NFAs were based on an earlier NFA that was developed by hand. In building the training set from an NFA the program would record its path as it randomly wandered through the NFA. Further details on the construction of the NFAs and the training set will be discussed later.

Overview

In training the neural network to recognize motion it becomes necessary to quantize motion into a form the neural network can be trained on. For these networks motion came to be described as a series of states, with an output value for the neural network being a value assigned to the transition between the states. Each state represented an NxN snapshot of motion. Building the entire NFA was deemed unfeasible, as just 9 inputs with 255 distinct values per input gives 4.56 x 10^{21} states, not counting all the transitions between the states. Thus it

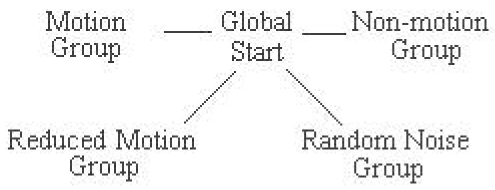

Fig. 1. Global breakdown of the NFA used to create the training set, note that each subgroup is an NFA as well.

becomes necessary to rely on the neural network's ability to generalize on input not seen during training, and use some subset of all the available states and transitions. The first subset was created by hand and consisted of an NFA with 141 states. An automatic method was then developed that would construct varying NFAs based around the NFA built by hand.

Design of the NFA

The NFA can be broken down into four main groups: motion, reduced motion, random noise, and non-motion. Each group is in turn broken down into sub-groups, each of which for the most part is an independent NFA. Due to space restrictions only the basis of design for the 3x3 training set will be shown in-depth with general differences of the 5x5 being explained in brief.

All four of the main groups can be accessed by and eventually lead back to the global start state as can be seen in Figure 1. This global start state consists solely of zeros. Any transitions to this state from any other state is assigned a weight of zero.

A more detailed look of the motion group can be seen in Figure 2. As this NFA is designed to impart the neural network with the preference to fire on upward motion, only transitions between states expressing motion in an upward direction are assigned a weight of 1, all other transitions are assigned a weight of 0. All motion states were created automatically by moving a line of varying width across the input field of view and linking successive states together. For the 5X5 network's NFA the 45 degree diagonal subgroups were replaced with 30 degree diagonal subgroups. The 5X5 network's NFA also had states in which motion over two time steps were connected via a transition to better aid the neural network in tracking a faster moving object.

The reduced motion group consisted of three subgroups, each of which looked like the subgroup shown in Figure 2 for the Up/Down motion. The difference is that instead of all intensities being set to 1, for each subgroup a different intensity value was assigned. Intensities of 0.5, 0.1, and 0.05 were used for both the 3x3 and the 5x5 networks. In addition the weights for motion in an upward direction were modified to values of 0.99, 0.1, and 0.0 respectively. Only up/down motion was dealt with, it was thought unnecessary to create similar NFAs for the diagonal or left-right motion as it was hoped the neural network would output a value of zero no matter the intensity of an object with such motion. These states were created by moving a line of varying width and intensity values across the input field of view in an upward, then downward direction. Successive states were linked together, and transition values assigned base on line direction and intensity.

The random noise group was built so that the neural network would be unresponsive to static noise. Each subgroup of the random noise group had its own individual start state that was basically an all-zero state. From each individual start state four random states within that subgroup were chosen to be reachable from that subgroup's start state. Each state within the same subgroup could reach some randomly selected four other states within that subgroup. In addition all states within that subgroup, except the start state, could reach the global start state. All transitions within a group of random states had a weight of zero and were unidirectional. All states within a subgroup, except the start state for that group, consisted of randomly determined input values that fell between the range of 0.0 and 1.0.

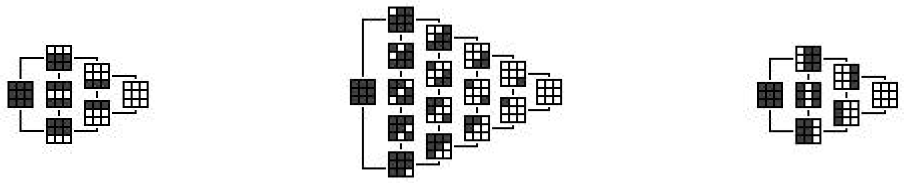
Fig. 2. From left to right are the NFA's for the motion subgroups: Up/Down, Up-Left/Down-Right, and Left/Right. Not pictured is the Up-Right/Down-Left subgroup. All connections are bi-directional. The state with all black boxes is the global start state, the state with all white boxes is the motion nexus state. The nexus state connects all the motion subgroups together.

The non-motion group was built so that the network would be unresponsive to an unmoving object in the field of view of the neuron. This was accomplished by generating duplicate states of the same input pattern. A set of four states was created and linked together as shown in Figure 3. A state was created by setting the value of each cell in the state between 0 and 1 randomly. Any value less than 0.5 was then set to a near zero value (< 0.001). A non-motion start state would be created that the global start state would connect to (in Figure 3 this is the second state from the left). The non-motion start state would be a state where all input values were zero and would connect to multiple sets of non-motion state sets. Usually there would be four non-motion state sets for the non-motion start state to connect to. Each state in a non-

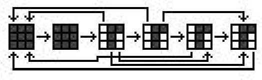
Fig. 3. Part of a Non-Motion subgroup. Of interest is the way the non-motion states are linked together.

motion state set would be able to reach the global start state, except for the non-motion start state.

Control Features

There are various control features that may be used to modify what sort of motion the NFA constructed will favor. At the moment these features are line rotation, movement rate, number of random subgroups, number of non-motion subgroups, and threshold cutoffs for the non-motion subgroup. The rate of rotation for the line can determine how invariant the network is concerning the preferred direction of motion. Movement rate, or moving the line more than once in a time step, is important for input fields of 5x5 or larger where objects have the potential for such motion across the input field. The number of random subgroups can determine how unresponsive a network will be to static noise. The number of non-motion subgroups affects how well the network will ignore stationary objects. The thresholds for cut off in the network can affect how sensitive the network is to the intensity difference between a moving object and its background.

Generating the Training Sequence

In generating a training sequence, the global start state was always the first state in the sequence. From there the rest of the states in the training sequence were generated by randomly following the transitions through the NFA. Additional modifications were done during the generation of the training sequence to give the neural network a broader input pattern set. This included variable background noise, slight variance of the intensity of the states in the motion group, allowing for delays between changes in states, and the number of state snap shots used. Last to be discussed is a response decay factor that could be used in tracking the direction of the motion on a global level.

To give the effect of background noise, values of states that were zero or close to zero (< 0.01) would be randomly reset to a value very close to zero (<0.009). In addition to varying the background noise when states from the motion group were used in the training sequence, their 1.0 intensity values were replaced with values between 0.5 and 1.0. No modified values between any two cells in such a state were more than 0.2 apart in intensity. To keep such mappings consistent with the apparent motion being generated, the intensity values would first be mapped to a 10x10 torus. Then, by looking through a NxN window of the torus and masking out any near zero values, the input states would be modified. The NxN window would be moved over the torus in the correct apparent direction each time step. A new torus was generated each time the NFA returned to the global start state. Lastly, states would be allowed to randomly stay in the same state for a random number of steps. This was to aid the neural network in ignoring stationary objects and moving objects coming to a stop. The overall size of the training sequence could be set to any number, the larger the number the better overall view the neural network would get of the state space. When the training pattern generator reached a given number of states it would continue until it reached the global start state at which point it would stop.

To aid in following motion direction on a global level, a response decay factor for the motion was incorporated into the training process by updating the transition weights on the fly. This means a moving object will leave a trail as it moves past a group of motion sensing neurons. Such a decay occurs when a state transitions from a preferred motion state to a state that generates a zero output. The rate of decay is exponential, such that within four time steps the output weight is back to zero. The rate of decay is reset whenever there is a transition to a state with a higher transition value than that of the decayed value or once the decayed value has gone below 0.2. For the former the higher transition value is used, if it is the latter then the decayed weight is set to zero.

A Jordan-Elman Network

A Jordan-Elman network is a recurrent network developed by Jordan that was modified to have short term memory (STM) by Elman with the addition of a context layer (Jordan 1986 and Elman 1990). The input layer of such a network is fully connected to the first hidden layer. Each hidden layer is in turn fully connected to the next hidden layer, with the last hidden layer being fully connected to the output layer. In addition each hidden layer has a context layer associated with it. Each node in a hidden layer has a 1-to-1 connection to its corresponding node in the context layer associated with that node's hidden layer. Each context layer is fully connected to the hidden layer to which it is associated and in addition each node in a context layer is connected to itself. The output layer may or may not have a context layer. The context layer acts to save the activation values of the hidden layer for one time step, and these values are used as additional input values to the hidden layer on the subsequent time step. This allows the network the ability to implement any arbitrary finite state automata (Minsky 1967).

Both the 3x3 network that was trained on the hand built NFA and the one that used the automatically built NFA consisted of 9 input states, 18 hidden states with their context layer counterparts, and one output state. The 5x5 network consisted of 25 input units, 30 hidden units with their context unit counterparts, and one output unit. Each network was trained on its training sequence for 200 to

300 iterations, at which point the mean squared error reached around 0.003. The 3x3 network using the hand built NFA used a 60,000 state training sequence, while the other two networks used only a 40,000 state training sequence. This was due mainly to storage limitations on the hard disk being used at the time.

Setting the Networks up for Testing

All the networks were trained to respond to upward motion only. To generate networks that would respond to motion in other directions the input field being sent to the network was merely rotated. This was done to save time on the number of Jordan-Elman networks that would

Fig. 4. Shows the layout for the small and medium filters for a 3x3 input field.

require training. This meant that for the network to notice downward motion the image field was rotated 180 degrees

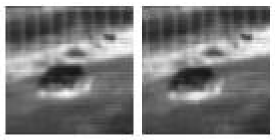

Fig. 5. Frames 13 and 14 from a 32 frame motion sequence.

so that if an object was moving down, it would appear to the network as if it were moving up. In this manner a total of 8 distinct direction sensitive (DS) neurons were created. For each direction 700 DS neurons were placed randomly throughout the image, for a total of 5600 neurons. It was possible for two different DS neurons to occupy the same point in the image.

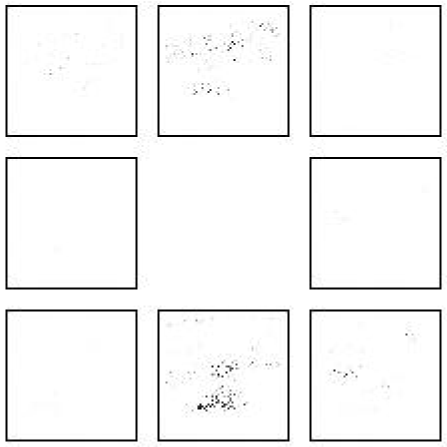

Fig. 6. Output for the 3x3 network based on the hand built NFA. Position in the figure represents the direction sensitivity for the neuron group.

To extend the range of the 3x3 and 5x5 networks 3 different filters were used that would give the networks input from a larger viewing area of the image. These three filters basically do an averaging over an area of the image. The smallest filter took 2x2 sets of pixels and averaged them to one value, while the largest took 4x4 sets of pixels and averaged them to one value. The image's field of view was split into 3 sections: a middle ring, an outer ring, and the main area in the center. A random mixture of all three filter types were placed in the center of the image space, while only the smallest filter size was placed within the inside outer ring. Base input field networks were placed on the outer most ring.

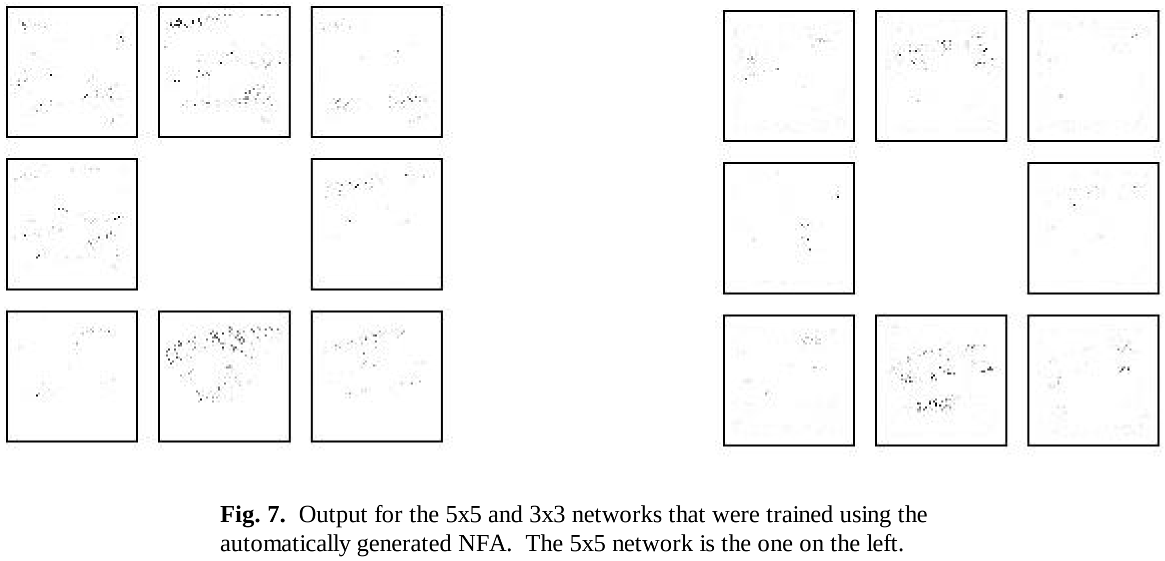

Fig. 7. Output for the 5x5 and 3x3 networks that were trained using the automatically generated NFA. The 5x5 network is the one on the left.

Results

Each network was run on a sequence of images that were taken with a thermal camera that slightly tracked a car moving in a down-left direction along a road. This means along with the down-left motion of the car there was apparent motion in the background in an up-right direction. Frames 13 and 14 from this sequence are shown in Figure 5. The outputs generated by the neural networks during this two frame sequence are shown in Figures 6 and 7. Each of the output figures have their images arranged based on the direction of motion sensitivity for the neurons. Thus the output for the down sensitive neurons is shown in the bottom middle image of each figure.

As can be seen the data for the 3x3 network that used the hand built NFA showed the best results, followed by the 5x5 network, and last was the 3x3 network that used the NFA built using the automatic method. Quality of the network's ability is shown in how well it picked up the car's motion and in ignoring the rest of the background. In this case the car's lower bumper had a local downward motion that was picked up well by the 5x5 network and the manual version of the 3x3. Also of note was the trail of motion left by the moving bumper in the manual version of the 3x3 that could be used to determine the true direction of the car. As for ignoring background noise, both of the automatically trained networks showed problems here, though some of it was due to the apparent motion.

Conclusions

While it is easy to see that the 3x3 network that was trained on the hand built NFA did better than both the networks that were trained on the automatically generated NFAs, it does not mean the automatic method is useless. The 5x5 did manage to come close to the hand built 3x3 network in performance. Through improvements in the building of the automatic NFA it is felt that such networks could come to equal the hand built 3x3 if only due to the fact that the NFAs will be identical in nature.

The lesser ability of the automatically trained networks can be attributed to three key areas of work: the non-motion subgroups, diagonal motion, and incorrect transitions. Of concern in the creation of the non-motion subgroups is ensuring that they don't contradict with states from the motion or reduced motion subgroups. Diagonal motion is a concern as well. If not done properly then the network may be too invariant in what sort of motion it will respond to. Finally, transitions need to be better policed to ensure there are no contradictions in the NFA, especially where the states dealing with the preferred motion are concerned. Making sure any transitions from a state to itself has an appropriate weight of zero for output value is one way to correct for this.

Lastly, two problems that don't necessarily deal with the creation of the NFAs directly, but do affect the performance of the networks are the compilation of the training sequences and the number of states that make up the training sequence. Of particular interest is how many states are needed to properly represent an NFA with X amount of states, and what sort of cross section of states from the NFA give the best results.

These changes would hopefully allow for easy and quick construction of variable training sequences for Jordan-Elman and/or other neural networks with similar properties. Such training sequences would allow for easy to develop motion neurons that could be customized for the problem at hand.

References

Borg-Graham, Lyle J. and Grzywacz, Norberto M. 1992. A Model of the Directional Selectivity Circuit in Retina: Transformations by Neurons Singly and in Concert. *Neural Nets: Foundations to Applications* 347-375.

Elman, J. 1990. Finding structure in time. *Cognitive Science* **14**, 179-211.

Hartline, Peter H., Kass, Leonard, Loop, Michael S. 1978. Merging of Modalities in the Optic Tectum: Infrared and Visual Integration in Rattlesnakes. *Science* 199:1225-1229.

Hunstberger, Terry. 1995. Biologically Motivated Cross-Modality Sensory Fusion System for Automatic Target Recognition. *Neural Networks* 8:1215-1226.

Jordan, M. J. 1986. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proc. Eighth Ann Conf. Of Cognitive Science Society,* 531-546.

Klauber, Laurence M. 1997. *Rattlesnakes.* Berkely, Calif.: Univ. of California Press.

Minsky, M. 1967. *Computation: Finite and Infinite Machines*. Englewood Cliffs, NJ: Prentice-Hall.

Missler, James M. and Kamangar, Farhad A. 1995. A Neural Network for Pursuit Tracking Inspired by the Fly Visual System. *Neural Networks* 8:463-480.

Newman, Eric A. and Hartline, Peter H. March 1982. Snakes of two families can detect and localize sources of infrared radiation. Infrared and visible-light information are integrated in the brain to yield a unique wide-spectrum picture of the world. *Scientific American* 246:116-127.

Rosenberg, Alexander F. and Ariel, Michael. May 1991. Electrophysiological Evidence for a Direct Projection of Direction-Sensitive Retinal Ganglion Cells to the Turtle's Accessory Optic System. *Journal of Neurophysiology* 65:1022-1033.

Reichert, Heinrich 1992. *Introduction to Neurobiology*. New York: Oxford University.