Real-Time Neural Network Processing of Gestural and Acoustic Signals

Michael Lee, Adrian Freed, David Wessel

Center for New Music and Audio Technologies (CNMAT)
University of California
1750 Arch Street, Berkeley, CA 94709
lee, adrian, wessel@cnmat.berkeley.edu

Abstract
We have added a new object to the MAX language to perform neural computations. By placing the neural processing in the context of a flexible real-time musical programming environment, we can conduct experiments rapidly on the application of the adaptive techniques to musical problems. We have trained the networks to recognize gestures from a MIDI keyboard, a Zeta™ MIDI guitar, radio drum, and Lightning™. In one experiment, a network successfully recognized musical motifs played on a MIDI guitar and issued messages to synchronize the presentation of images from a HyperMedia work. This "page turner" generalized important features of the motifs from 9 training sets. We have also trained a neural network to recognize gestural contours from the continuous spatial controllers, suggesting their application in the interpretation of conducting gestures. We discuss the low-level features extracted for the networks to operate on, how we trained the networks and musical applications of adaptive techniques.

Introduction
MAX, a real-time musical programming environment, was developed at IRCAM by Miller Puckette [PuZi90]. Users can assemble MAX programs by graphically patching together code or objects. The MAX environment allows users to extend the language by integrating external code resources, or objects, into the system much like HyperCard XCMDs [Zica90]. This combination of attributes makes MAX a flexible environment well suited for prototyping of real-time systems.

We have added a new object to MAX, MAXNet, which performs neural network computations. MAXNet can simulate multi-layered feed forward and Jordan [Jord86] and Elman [Elma88] style recurrent networks. By placing the neural processing in a flexible real-time musical programming environment, we can conduct experiments rapidly on the application of the adaptive techniques to musical problems.

Neural Networks
Neural networks are computational systems modelled after the architecture of the brain [RuMc86]. The brain is composed of an array of interconnected processing elements, or neurons. In a neural network, each neuron is a simple element that feeds the weighted sum of its inputs into a 'squashing' function such as a sigmoid. The value of a weight represents the connection strength between two neurons. For example, w_{ij} represents the connection strength from neuron i to neuron j.

$$f(net_j) = 1/(1+e^{-net_j * slopeFactor_j}) \quad (1)$$

$$\text{where } net_j = \sum w_{ij} o_i \quad (2)$$

Neural networks can perform attribute estimation, or function approximation, as well as classification. Researchers have proven that 3 layer neural networks can approximate any arbitrary mapping [HoSW89]. Attribute estimation is important because in many control applications, attribute values can be equally as important as class membership. For example, in the popular cart and pole balancing control problem [MiSW90], the neural network is required to produce control values for the cart and not a classification. (Classification can be thought of as a degenerate case of estimation [Pao89].)

Training a neural network amounts to searching for a set of weights that will give the neural network the desired input/output behavior. The most popular technique for training multi-layered feed forward neural networks is back-propagation gradient descent [RuMc86]. Back-propagation is an example of supervised training: the network is repeatedly shown both the input and its associated output pattern. After each presentation, the neural network tries to adapt its interconnection weights according to some cost function (this is sometimes referred to as on-line learning. One pass through all training examples is called an epoch). The process continues until some error criterion is satisfied.

MAXNet
MAXNet is a neural network simulator that can simulate multi-layered feed forward and Jordan and Elman style recurrent networks. The user can specify several architectural parameters: number of input and output neurons, number of neurons per

hidden layer, number of hidden layers, slope factors. Each layer is fully interconnected to adjacent layers, however "unconnected" links can be represented by weights of value 0.

Jordan and Elman recurrent networks are similar to strictly feed forward neural networks. Both style recurrent nets have context neurons that get their input from either a hidden (Elman) or output (Jordan) neuron. The extra context units are like input units, which get their inputs from other neurons instead of an external stimulus. This can be achieved by configuring a network with the number of input neurons equal to the number of external stimuli sources plus the number of output or hidden neurons (depending on recurrent style). The modified input patterns will be a vector consisting of the normal stimulus vector prepended to a vector describing the feedback paths. (See [Todd89] for musical applications of recurrent neural networks.) Another external control adjusts the amount of feedback from a context neuron to itself (this gives the neuron some memory).

The slope parameter also can be set for all output layer units. Setting the parameter less than 1 will decrease the slope of the sigmoid and make the function behave like a linear unit. Higher slope values seem to speed convergence of neural networks applied to problems with binary valued outputs.

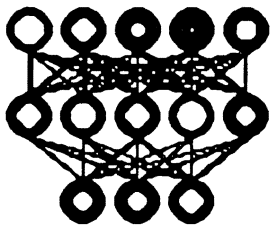

Figure 1. MAXNet Graphic Display

There is a companion program, MACNet, which runs on Macintosh and UNIX machines to facilitate training. Both code objects have identical capabilities and read and write the same format text files. The most efficient way to use MAXNet is to setup a data acquisition harness in MAX and collect the data into a training file. Then use MACNet to learn the input/output mapping and save the weights file. Finally, attach MAXNet to the acquisition harness, read in the weights, and then compute. Both simulators use compatible text files, which eases examining and editing weights.

MAXNet also can graphically display the state of the network in real–time. This feature can be turned on or off by the user. MAXNet uses color to represent the weight values and the size of the hole in the middle of a neuron to represent its activation level (see Figure 1). Connecting MAX user interface objects, such as sliders, to MAXNet gives you the ability to explore how the activation levels of input neurons affect the neurons in other layers.

Training the Networks

MAXNet uses several user adjustable parameters during a training session. The first two are learning rate and momentum, which are used for adapting the weights of the network. MAXNet also can learn the slope factor for individual output units. This requires that the user set a slope learning rate and slope momentum in addition to the weight adjustment parameters (note that a slope learning rate of 0 is conventional back–propagation). The generalized update equations for weights and slope factors is shown in (3) and (4).

$$\Delta x_{n+1} = \text{learningRate} * -\partial\text{Cost}/\partial x + \text{momentum} * \Delta x_n \quad (3)$$

$$x_{n+1} = x_n + \Delta x_{n+1} \quad (4)$$

MAXNet also can optionally add hidden units according to some error criteria. This means that the number of hidden units does not need to be determined *apriori*. A new node can be added if, for example the error has not decreased more than some percentage over the last n epochs [HiYH91]. The parameters the user can set are maximum number of hidden units (to terminate learning), percentage of error decrease, and number of epochs to train before applying grow criteria.

Real–Time Processing Requirements

Training MAXNet is computationally intensive, but once the weights have been found, running MAXNet feed forward requires less computing resources. There is a multiply/add for each weight and a function evaluation for each hidden and output neuron. For large nets, the number of connections exceeds the number of neurons so we can assume that most of the time will be spent computing weighted sums. On a 16MHz 68020 without a math coprocessor, we can achieve at least 200K connections/Sec. This translates to one forward pass through a three layer net with 10 neurons on each layer in 1 mSec. We are porting the MAXNet to a separate 68020 processor card,

the GreenSprings 1260, to reduce the processing requirements of the host 68020 and guarantee real-time.

Applications

We have trained the neural networks to recognize gestures from a MIDI keyboard, a Zeta™ MIDI guitar, radio drum, and Lightning™. We have also trained a neural network to recognize gestural contours from the continuous spatial controllers, suggesting their application in the interpretation of conducting gestures. With the recurrent capabilities of MAXNet, algorithmic composition in the manner of Todd [Todd89] is also possible.

MIDI Keyboard

MAXNet can construct complex multidimensional maps. One application is to dynamically control the timbre using a combination of key number, velocity and another MIDI controller signal to control the mix of a bank of tone generators. With a neural network, complex regions can be carved out of the key, velocity, controller space to represent certain timbres.

This type of control also can be achieved by using multidimensional tables or functions. However, neural networks have the advantage over tables that they can interpolate points and generalize; i.e., the whole table does not need to be filled out. Multivariable functions could be computationally more efficient than neural networks, but it is much easier to find an approximating function by training a neural network. Training a neural network to approximate multivariable functions automates the search and is ultimately less time consuming for the human.

Radio Drum

The radio drum problem represents a coordinate transformation application of a neural network. The drum, developed at Bell Labs by Mathews and Boie [MaSc89], has two sticks, each of which has a transmitter attached to the end, and a surface with an array of antennas. Each transmitter operates at a different frequency and the antennas report the intensity of the transmitter signal. The response of the stick antenna combination is highly nonlinear. In addition, when the two sticks are close together, a proximity effect decreases the antenna readings.

The initial goal was to feed the antenna readings directly into the neural network and transform them into 3D spatial coordinates compensating for the proximity effects along the way. Once the neural network was trained, the resulting system behaved more linearly because the neural network removed the nonlinearities.

With the neural network, the 3D coordinate transformation could be bypassed and the antenna space could be directly translated into the gesture space. This would eliminate the need for further computations that take the gesture from 3D space into gesture space, thus possibly reducing errors.

MIDI Guitar

In another experiment, a neural network successfully recognized musical motifs played on a MIDI guitar and issued messages to synchronize the presentation of images from a HyperMedia work. The guitar provided MIDI data, which was analyzed and provided the tonal center, note density, note variance, duration measure, and melodic contour for each string. The riffs were segmented heuristically by setting a time threshold between events. The neural network was trained and learned to generalize these feature vectors from 9 different motifs.The resulting net had 37 input units (one extra set for global features), 6 hidden units, and 9 output units.

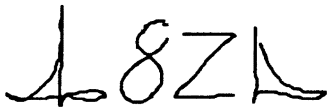

Figure 3. Shapes Recognized by MAXNet

Lightning™

Lightning™, a new controller developed by Don Buchla, has two infrared transmitters and a receiver box, which provides spatial coordinates of the transmitters. We have interfaced Lightning™ to a neural network to recognize conducting gestures (see Figure 3). Each gesture was analyzed with a technique developed by You and Fu for curve attributes and syntactic pattern recognition [YoFu82] and provided a curvature, symmetry, and endpoint distance measure. These features were normalized and then presented to the neural network. After training, the neural network could classify unseen gestures according to the training sets.

Discussion

Neural networks seem well suited to musical applications due to their adaptive nature. We can now develop experimental instruments that learn

the musician's personal mapping between gesture and response. This adaptive approach to instrumental control could be applied to more traditional instrument controllers as well. For example, a guitar controller could be self–adjusting to a particular playing style. A training set consisting of specified scales and chords could be described in the instruction manual and once the musician performed these examples the instrument would adapt itself automatically. Other applications might be to use them to compensate for physical disabilities or to create more ergonomic instruments.

An intriguing question arises around the notion of adaptive instruments. How will we rig the training harnesses so as to best exploit the very adaptive nature of the human performer and an instrument that learns?

Further Research

In gesture recognition for music applications such as conducting or instrument control, segmentation, time warping, and other timing issues, will play critical roles. The continuous nature of these applications demand that a complete time representation be formulated.

Time–Delay Neural Net (TDNN) architectures in speech and handwriting recognition applications [GHAL91] [WHHS89] address some timing issues and offer possible solutions. Several researchers [ToPe89] have also reported success with sequence recognition using self–organizing neural networks. Both techniques need to be further investigated.

This work was supported in part by grant INV 9004-011 from Apple Computer Inc.

References

[Elma88] Elman, J.L., "Finding Structure in Time." Technical Report 8801: La Jolla: University of California at San Diego, Center for Research in Language.

[GHAL91] Guyon, I., Henderson, D., Albrecht, P., Le Cun, Y., Denker, J., "Writer Independent and Writer Adaptive Neural Network for On–line Character Recognition." To appear in IWFHR–2, 1991.

[HiYH91] Hirose, Y., Yamashita, K., Hijiya, S., "Back Propagation Algorithm Which Varies the Number of Hidden Units." *Neural Networks*, Vol. 4, 1991.

[HoSW89] Hornik, K., Stinchcombe, M., White, H., "Multilayer Feedforward Networks are Universal Approximators." *Neural Networks*, Vol. 4, 1989.

[Jord86] Jordan, M.I. "Serial Order: A Parallel Distributed Processing Approach." Technical Report ICS–8604. La Jolla: University of California at San Diego, Institute for Cognitive Science, 1986.

[MaSc89] Mathews, M., Schloss, A., "The Radio Drum as a Synthesizer Controller," Proceedings of the ICMC, 1989.

[MiSW90] Miller, W.T., Sutton, R.S., Werbos, P.J., eds., *Neural Networks for Control*, MIT Press, Cambridge, MA, 1990.

[Pao89] Pao, Y.H. *Adaptive Pattern Recognition and Neural Networks*, Addison–Wesley, Reading, MA, 1989.

[PuZi90] Puckette, M., Zicarelli, D., *MAX – An Interactive Graphic Programming Environment*, Opcode Systems, Menlo Park, CA, 1990.

[RuMc86] Rumelhart, D.E., McClelland, J.L. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vols. 1 and 2, MIT Press, Cambridge, MA, 1986.

[Todd89] Todd, P., "A Connectionist Approach to Algorithmic Composition," Computer Music Journal, Vol. 13, no. 4, 1989.

[ToPe89] Tolat, V.V., Peterson, A.M., "A Self Organizing Neural Network for Classifying Sequences." Proceedings of the 2nd IJCNN, Vol. 2, 1989.

[WHHS89] Waibel, A., Hanazawa, T. Hinton, G., Shikano, K., and Lang, K.J."Phoneme Recognition Using Time–Delay Neural Networks," IEEE Transactions on Acoustics, Speech, and Signal Procesing, 37:328-339, March 1989.

[YoFu79] You, K.C., Fu, K.S., "A Syntactic Approach to Shape Recognition Using Attributed Grammars," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC–9, No. 6, 1979.

[Zica90] Zicarelli, D. "Writing External Objects for MAX," Opcode Systems, Menlo Park, CA, 1990.