Int. J. Engng Ed. Vol. 21, No. 4, pp. 636–648, 2005
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 2005 TEMPUS Publications.

MATLAB-based Introduction to Neural Networks for Sensors Curriculum*

ROHIT DUA, STEVE E. WATKINS, SAMUEL A. MULDER and DONALD C. WUNSCH
Department of Electrical and Computer Engineering, University of Missouri-Rolla, Rolla, MO 65409, USA. E-mail: rdua@umr.edu

Specialists and non-specialists in artificial neural networks (ANN) must closely interact in many applications, including structural sensing. The non-specialists must be aware of ANN-specific terminology, capabilities, and connecting concepts for effective collaboration. An instructional approach for ANNs is described that progresses from practical concepts to guided MatLab-based experimentation. Back propagation-trained multilayer perceptron neural networks are presented with an emphasis on parallel processing and training characteristics. The one-week instructional module has a lecture to convey terminology and structure, detailed examples to illustrate the training process, and guided application-based exercises. The MatLab neural-networks toolbox provides a transparent learning environment in which the students focus on network design and training concepts rather than the tool itself. Learning effectiveness was evaluated in an applications-oriented sensors curriculum. Instructional resources including realistic problems are web-accessible. These resources may be adjusted for different degrees of challenge and for simpler or more realistic problem solving.

INTRODUCTION

ARTIFICIAL NEURAL NETWORKS (ANNs) are versatile tools that are often integrated with other technologies. In structural sensing applications, interdisciplinary considerations involving instrumentation, signal processing, materials, manufacturing, and structural mechanics determine the architecture and characteristics of the neural network implementation. Even for instrumentation teams concerned only with the electrical engineering issues, ANN specialists and non-specialists must work together closely. The non-specialist in ANN needs to be aware of discipline-specific terminology, ANN capabilities, and connecting concepts for effective collaboration. Consequently, tailored instruction that can quickly teach a systems-level understanding of ANN design and operation is needed. Such a qualitative introduction can supplement courses on sensors, instrumentation, control, and smart structures. MatLab-based exercises with non-trivial architecture and training can provide a key link to applications.

Traditional instruction in ANNs is typically through a comprehensive course at the graduate or upper undergraduate levels. Textbooks usually introduce algorithms and training functions with assumptions of some mathematical maturity [1]. While this approach may be appropriate for students who desire to be ANN specialists, other students can struggle with the mathematical detail. Also, these students may not have the time for a standalone course and may not be motivated without a connection to realistic applications. However, the concepts behind neural networks are fundamentally simple and have value for a more general audiences. Our research at the University of Missouri–Rolla in the smart structures area [2] involves monitoring structures using integral sensors and dedicated processing [3–5]. The instructional motivation and ANN target concepts grew out of our collaborative experience as researchers and student advisors [6]. Tailored student-centered learning [7] coupled with powerful software tools can provide benefits to students with and without prior experience with ANNs. Typically, the former better appreciate the relevance of ANNs to other fields while the latter develop collaborative knowledge.

This work describes a systems-level approach. The operation of ANNs is taught by linking foundational knowledge to a realistic context [8, 9]. The intended audience is composed of non-specialists who need to collaborate with specialists and to understand the capabilities of ANNs. An instructional module is proposed that progresses from practical concepts to guided MatLab-based experimentation. Instructional resources are made available on a course website. These resources may be adjusted for different degrees of challenge and for simpler or more realistic problem solving.

The learning objectives and stages are:

- to convey knowledge of terminology and structure;
- to illustrate the training process;
- to link ANNs to applications.

Back propagation-trained multilayer perceptron neural networks are emphasized due to their

* Accepted February 2005.

importance and problem-solving flexibility. The lecture defines key ANN components and intuitive explanations with minimal mathematics. The illustrations give step-by-step training results for simple network examples for approximating a mathematical function, identifying damage strain signatures, and predicting wing performance. The laboratory activity has guided application-based exercises, which allow students great flexibility in network design and in training parameters. There is not one correct answer. Exercise constraints allow choices that will produce non-convergent, poorly convergent, and well-convergent solutions. The MatLab neural-network toolbox with the graphical user interface (GUI) [10] provides a relatively transparent learning environment in which the students focus on network design and training concepts rather than the tool itself. The methodology was implemented and assessed for an applications-oriented collaborative course in the topical area of sensors and structural sensing [11, 12]. This paper discusses the design of the instructional module, documents the web-accessible resources, presents the qualitative examples and the MatLab exercise, and evaluates the course implementation.

INSTRUCTIONAL OVERVIEW

Methodology

The progression for the instructional module starts with a lecture conveying target knowledge and examples and ends with an interactive design and training exercise. The educational approach incorporates classic hearing, seeing, and doing components. The main goal is to teach a systems-level, applications-oriented understanding of artificial neural networks. A succinct presentation is desired for qualitative understanding. The instruction should be sufficient to be a standalone introduction for ANNs or to be a systems view that complements other ANN studies. Tools for collaboration are desired. ANN capabilities should be linked to applications and the module should be easily adjusted for needs in different application

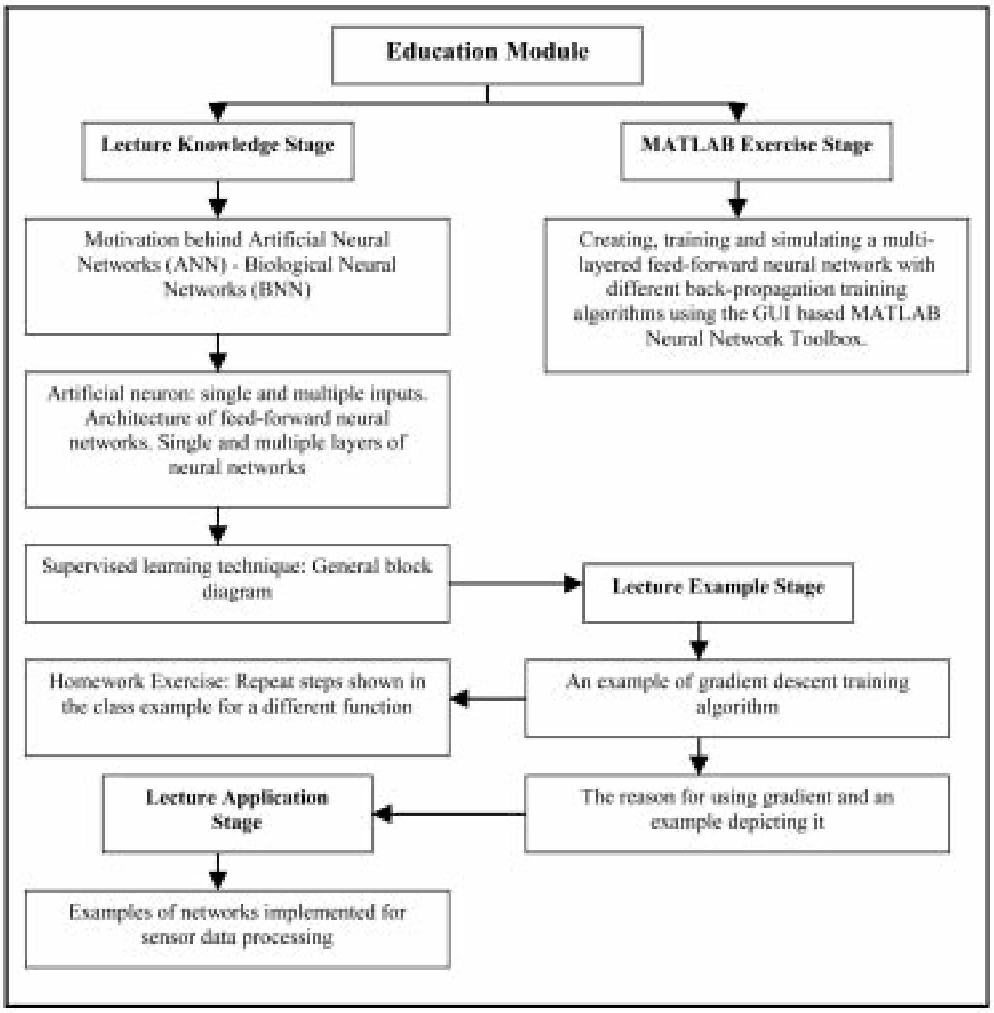

Education Module
Lecture Knowledge Stage
MATLAB Exercise Stage
Motivation behind Artificial Neural Networks (ANN) - Biological Neural Networks (BNN)
Creating, training and simulating a multi-layered feed-forward neural network with different back-propagation training algorithms using the GUI based MATLAB Neural Network Toolbox.
Artificial neuron: single and multiple inputs. Architecture of feed-forward neural networks. Single and multiple layers of neural networks
Supervised learning technique: General block diagram
Lecture Example Stage
Homework Exercise: Repeat steps shown in the class example for a different function
An example of gradient descent training algorithm
Lecture Application Stage
The reason for using gradient and an example depicting it
Examples of networks implemented for sensor data processing

Fig. 1. Flowchart depicting the progression of the module, which includes the lecture and MatLab exercise stages. The three stages of the lecture are: (1) to present terminology and structure (lecture knowledge stage), (2) to illustrate the training process (lecture example and homework stage), and (3) to link ANNs to applications.

areas, e.g. sensors. Minimal mathematics is desired so that the target audience can be general. The instruction is not intended to teach students to apply ANNs independently, but to build a capability to interact with ANN specialists on collaborative teams. Figure 1 shows the structure of the educational module. The lecture concepts are reinforced through examples with associated homework assignments and a capstone exercise using MatLab.

The ANN non-specialist needs an awareness of discipline-specific terminology, ANN capabilities, and connecting concepts for effective collaboration. Students should gain exposure to the target concepts of:

- terminology, including supervised learning, training cycles, convergence, and error;
- ANN architecture of layers, neurons, weights, and transfer functions;
- performance factors of network complexity, learning rate, and gradients;
- parallel processing of multiple inputs and outputs;
- capabilities for function approximations and data classification;
- problem definition for applications.

A qualitative understanding is sought for the single versatile case of multilayered-perceptrons feedforward networks with the back propagation training algorithm.

The learning environment for the laboratory exercise is MatLab with the neural network toolbox. This toolbox contains menus for common algorithms to pre- and post-process data, for setting up neural network architectures, and for operations and error display during training. In addition, the general computing resource can handle multiple-dimensional files as needed for input and output manipulation in many applications. Many computer languages are suitable for neural network implementation, but the GUI-based control of the MatLab toolbox facilitates this introductory exercise. No programming is needed and step-by-step visual procedures for setting up architectures and algorithms are possible. Intuitive control is critical for a non-specialist target audience.

Public access to module

The lecture notes including the examples, a multiple-choice quiz, the exercise assignments, MatLab data files, and other resources are available at the 'Educational Resources' section of the web site http://campus.umr.edu/smarteng/ [13]. The material was developed with MatLab v6.0, and MatLab neural network toolbox v4.0. For those instructors using the module, an assessment is requested. The questionnaire is included on the website. The instructional module is intended for upper-level undergraduates and beginning graduate students in electrical and computer engineering. The materials assume a working understanding of MatLab, but prior experience with the neural network toolbox and MatLab programming are not necessary. Other examples and exercises are planned which highlight additional applications.

GUIDED INSTRUCTION

Lecture

Multilayer, feedforward networks with various back propagation training algorithms are discussed exclusively in the lecture due to their practical importance and wide use [1]. The lecture topics are

- motivation of biological neural networks;
- the artificial neuron;
- the sigmoid and linear transfer functions;
- types of artificial neural networks;
- supervised learning techniques;
- back propagation training;
- training algorithms;
- function approximation examples;
- engineering examples of ANNs.

The lecture PowerPoint file, as given on the web site, is designed for a presentation between one and two hours. The presentation uses practical definitions, block diagrams, simple figures, and step-by-step procedures.

Lecture examples

Key features of the lecture are the examples. Four examples are included. The initial two examples show the details of the training process and emphasize the gradient procedure for convergence and minimizing error. The latter examples show how ANNs can provide engineering solutions.

The target concepts for the initial examples are network training details to obtain satisfactory weights for the neurons. Given a training set of input data with known outputs, the selected network with arbitrary weights produces incorrect outputs. The error is back propagated through the network and the weights are updated. In successful training, the neuron weights converge to values for which the network output has a prescribed error. The examples illustrate the effect of varying the learning rate from zero to one and of error rate-of-change on the rate of change for weights. The optimal learning rate is dependent on the specific problem and can achieve faster or slower convergence.

The initial examples are function approximation problems. These first highlight the training iterations so that error is minimized. In this case, the function relates the output y with a single input x and the mathematics are trivial, i.e. $y = x$. A network with an input layer, one hidden layer, one output layer, and the gradient descent training algorithm is used. A Sigmoid transfer function is used for neurons in the first and second layers and

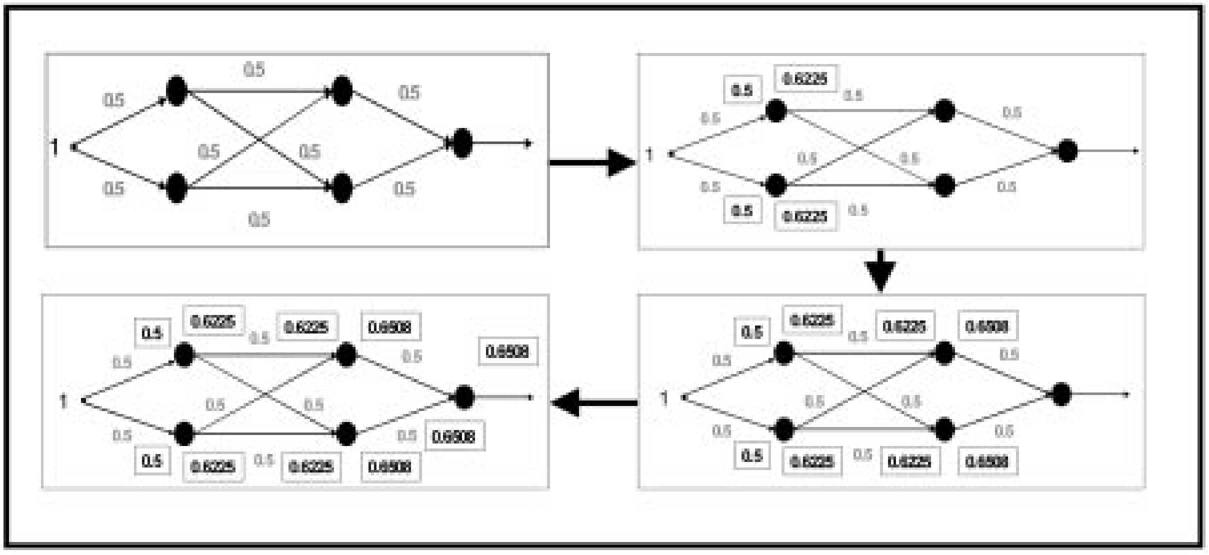
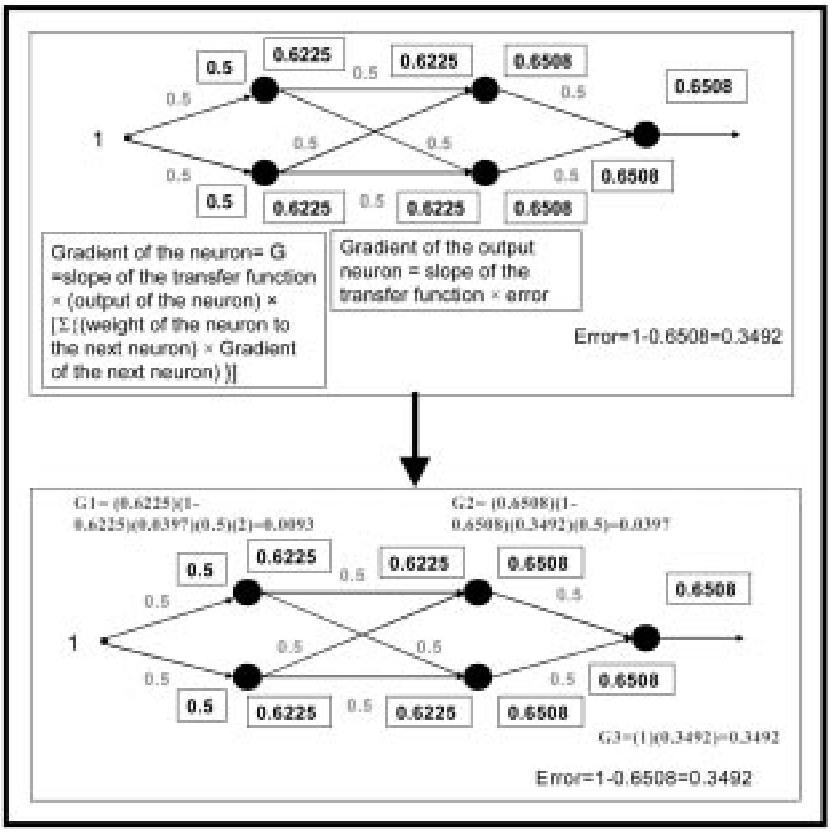


Fig. 2. First pass for function approximation ANN. The numerical values for the first training cycle are progressively shown.


1
0.5
0.6225
0.6508
Gradient of the neuron= G =slope of the transfer function × (output of the neuron) × [Σ{(weight of the neuron to the next neuron) × Gradient of the next neuron)}]
Gradient of the output neuron = slope of the transfer function × error
Error=1-0.6508=0.3492
G1= (0.6225)(1-0.6225)(0.0397)(0.5)(2)=0.0093
G2= (0.6508)(1-0.6508)(0.3492)(0.5)=0.0397
G3=(1)(0.3492)=0.3492
Error=1-0.6508=0.3492
Fig. 3. Error correction for ANN. The weight gradients are determined based on the output error through a back propagation procedure.

a linear transfer function is used for the output neuron. Figure 2 illustrates the first pass of the training sequence with arbitrary initial weights. Intermediate values and weights are explicitly shown. Figure 3 illustrates the determination of weight gradients. Note that the simulated output is subtracted from the expected output to obtain the error. Figure 4 shows the new weights and the second pass. A third pass is illustrated in the lecture. The symmetric values reinforce the steps in the process.

The second function approximation highlights the learning rate and shows multiple inputs and outputs. The function relates the outputs (1, 0) to the inputs (1, 0.4). This example was chosen to show the philosophy behind weight update, i.e. by how much and in which direction the weights are updated depending on value and sign of the error

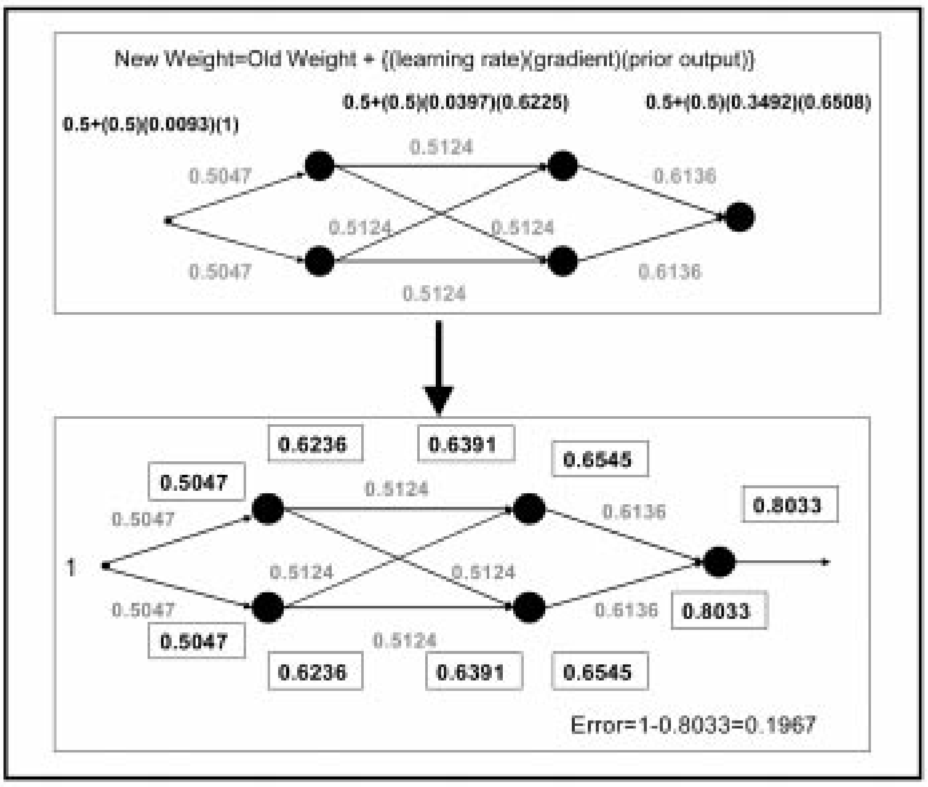


Fig. 4. Weight updates and second pass for function approximation ANN. The new weights produce a converging output.

generated. A network with one hidden, one output layer, and the gradient descent training algorithm is used. A Sigmoid transfer function is used for all neurons. Figure 5 illustrates the first pass of the training sequence and the gradient determination. The value of the gradients generated for the output layer decides how the weights will be updated as shown by dotted arrows in the figure. The initial weights are uniform and the asymmetry of the solution is readily apparent. This behavior leads to a lecture discussion of ANN performance.

A homework was assigned to get students to go through the step-by-step process of understanding the gradient descent training algorithm. Students were required to repeat the steps shown in the class example (Figs 2 and 3) for a different function (output $y = 3 \times$ input x). To keep the solutions consistent, an input of '1' was assigned to be used as the training and test input.

Two supplemental examples are included to connect the ANN concepts to sensor applications. The first is an identification problem for characterizing strain signatures during damage events using multilayered feedforward ANNs as classifiers. The second is a prediction problem for predicting strain in a wing based on airspeed and angle-of-attack information using ANNs as function approximators. Both the applications were projects carried out here at UMR and have been presented in major neural network conferences [4, 14]. In both the examples, students were shown generation of data for the respective problem, setup of the input vectors, the type of pre-processing done on the inputs, type of outputs (output classes for classification problem), architecture of the ANN used, training algorithm used to train the network and results from simulation of test data. Figure 6 shows how the error changes as training progresses. The display is a screen shot from the MatLab neural network toolbox to acquaint the students with the environment used in the exercise.

MatLab exercise

A MatLab-based laboratory exercise follows the lecture and homework assignment. This guided activity gives the students a wide range of design and training choices. Students are encouraged to start with simple feedforward networks and progress to more complex networks until a prescribed mean-squared-error condition is met. The assignment is not unconstrained. The training and testing data sets are preprocessed, a maximum of three layers is allowed, and only the LOGSIG [10] transfer function is used. The solution to the exercise problem is not unique; many design combinations can satisfy the error criteria, albeit at different rates and with different minimum values. However, many other design combinations will not satisfy the error criteria or will converge too slowly. Performance can be measured in terms of network simplicity (number of neurons and layers), minimum error, training cycles, and errors in the test simulation.

The exercise incorporates the IRIS-flower data set [15], which is a benchmark data classification

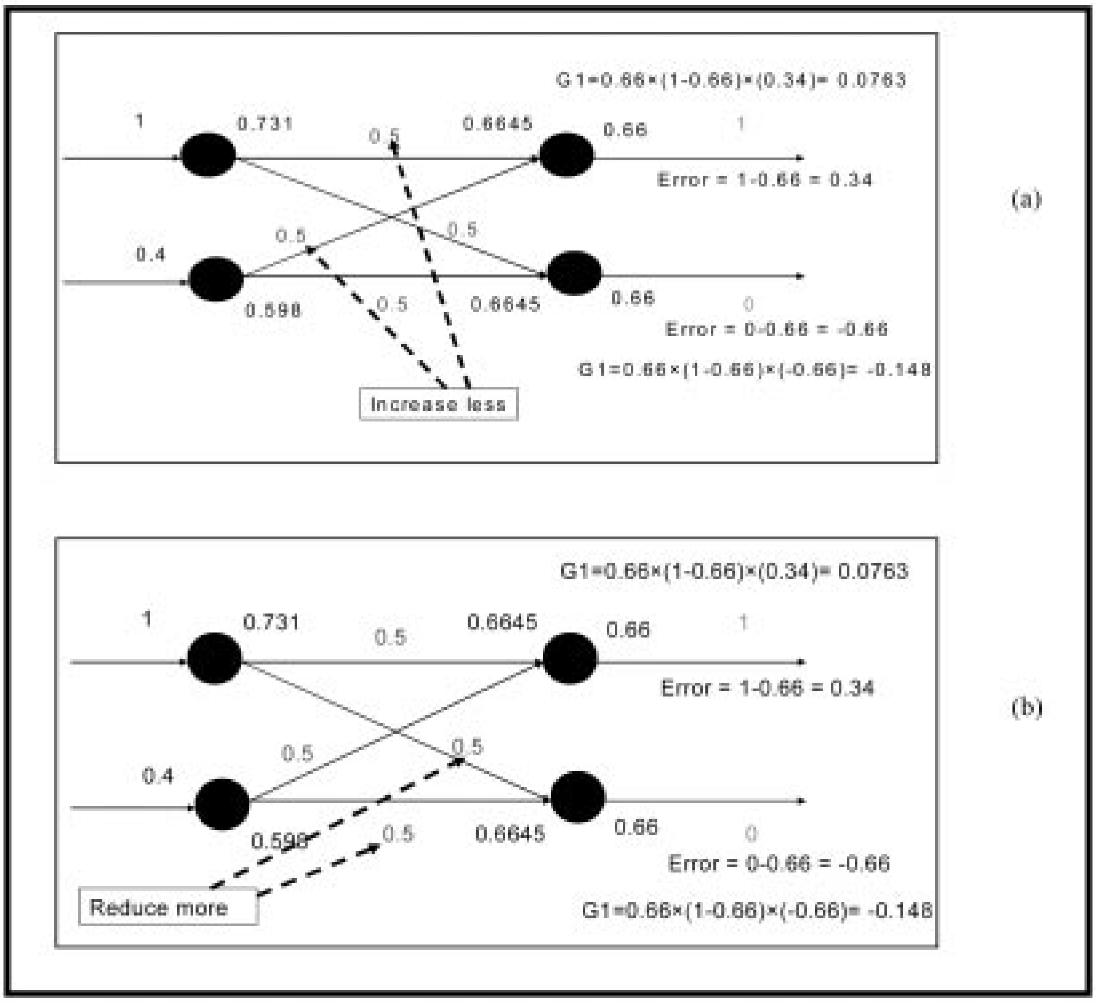


Fig. 5. Two-input/two-output ANN. The gradients decide the direction and value of the weight updates as shown in (a) and (b).

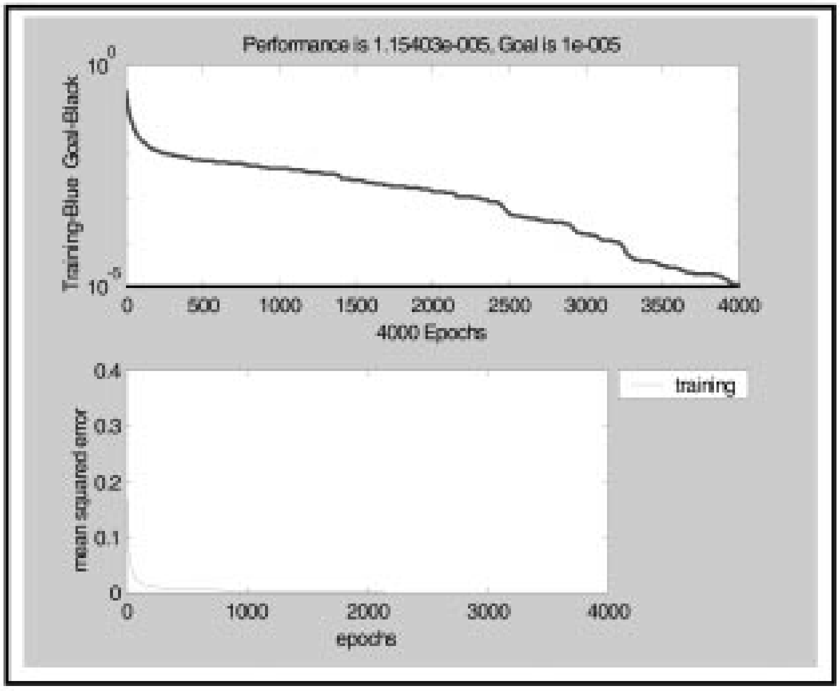


Fig. 6. Sample variation of mean-squared error as a function of training cycles, from the classification example.

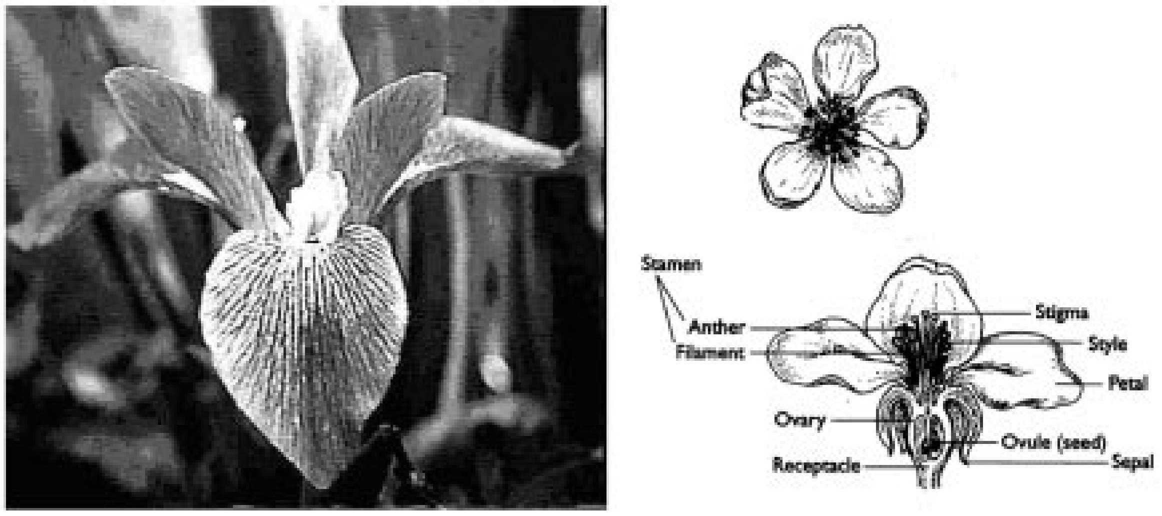


Fig. 7. *Iris verginica* (left). Note the parts of the flower. Used with permission from Brooklyn Botanic Garden [15].

problem. The network must distinguish among three types of similar flowers from the iris family, which are *iris setosa, iris verginica* and *iris verico-lor*. The distinguishing features are petal width, petal length, sepal width, and sepal length as shown in Fig. 7. The input data sets are pre-normalized as required by the ANN and ready for MatLab importation. The output classification is done using a binary code: Code 00 for Setosa, Code 01 for Verginica, and Code 10 for Versicolor. Consequently, the ANN must have four inputs and two binary outputs. Available training vector sets number 120 and testing vector sets number 30.

The MatLab neural network toolbox provides intuitive control through the GUI 'Network/Data Manager' windows. Figure 8 shows the options within the toolbox. Note the buttons for building the network architecture and importing the training and testing data. The 'new network' window gives menus for the student to select the type of network as well as the number of layers, number of neurons in each layer, the function for each neuron, the training algorithm, and learning rate. Other windows visually represent the location of variables and the network diagram. The network is defined and a pictorial view is available in the 'Create New Network' and 'Networks' window, see Fig. 9. The diagram specifies the number of layers, the neuron transfer functions, the number of inputs and the weight matrices. After the weights are initialized and the number of training cycles (epochs) is chosen, the ANN is ready for training, i.e. adjusting weights according to the error and the algorithm. Figure 10 shows the

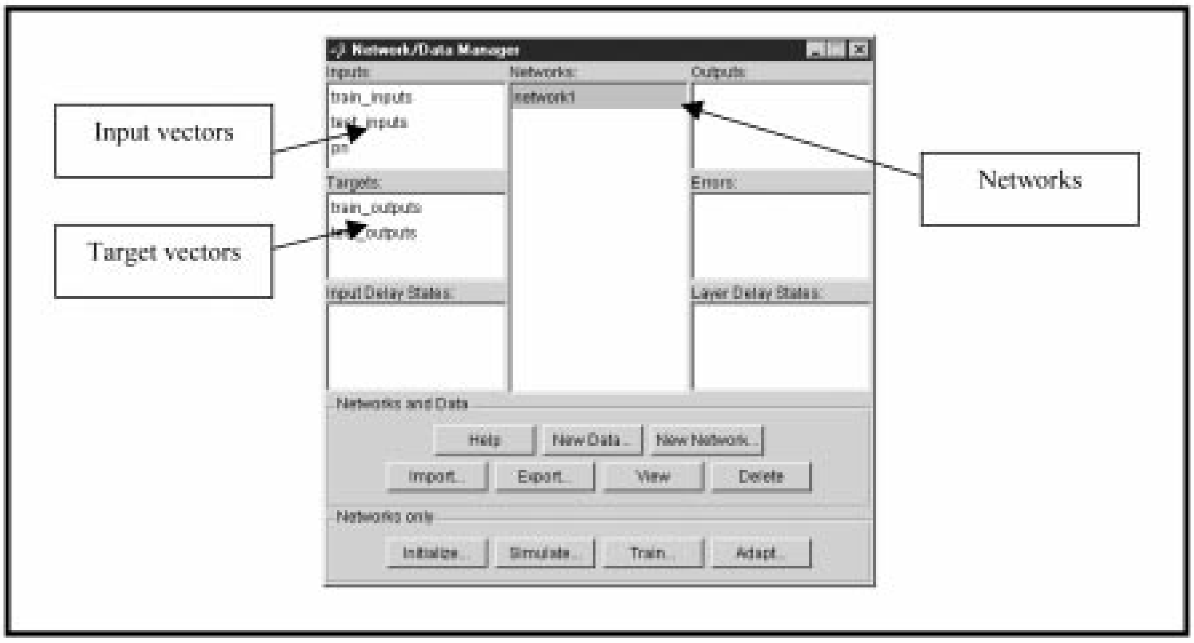
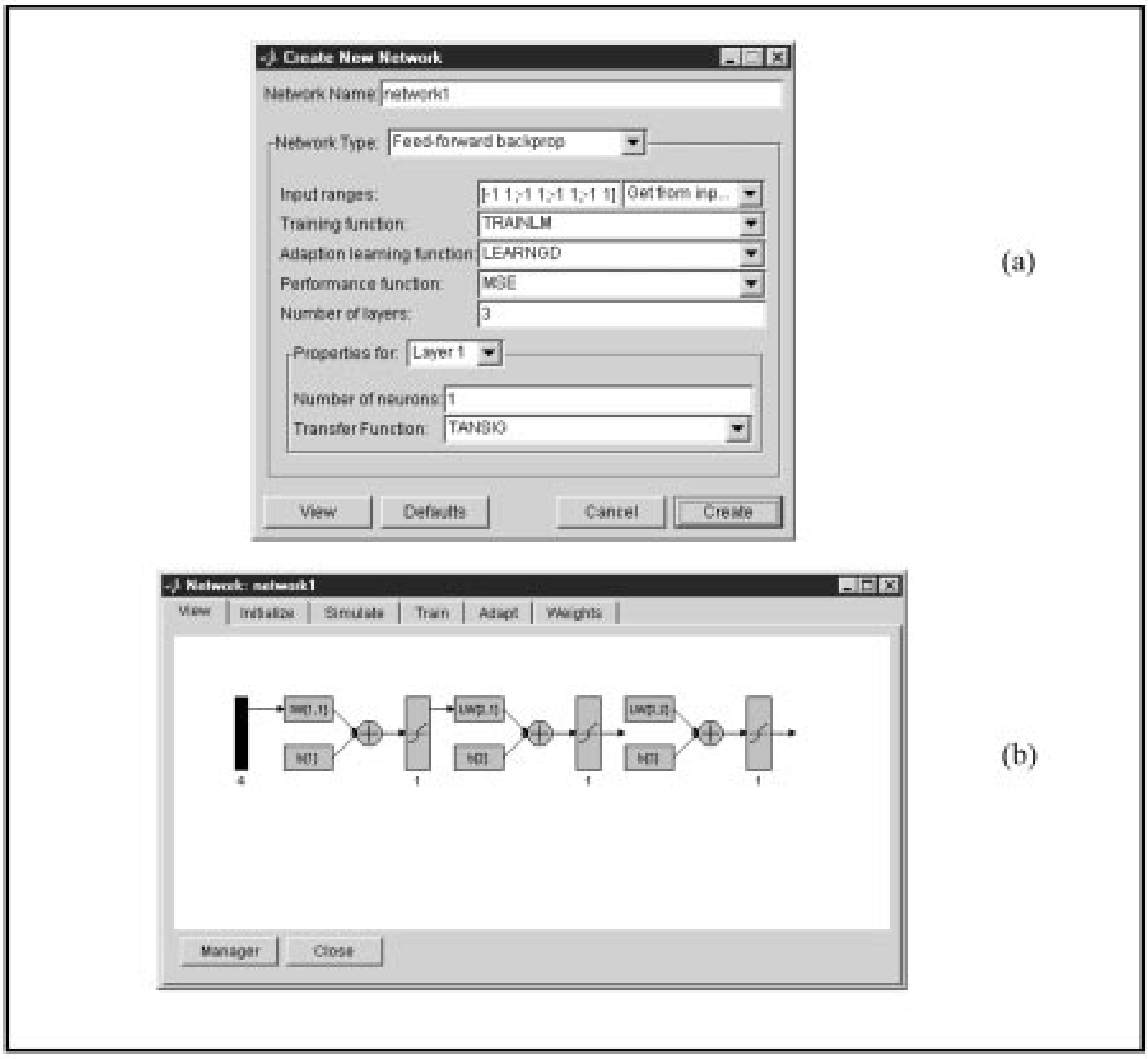


Fig. 8. The GUI-based neural network toolbox. The various buttons lead to options for building the network and handling the data.


Create New Network

Network Name: network1

Network Type: Feed-forward backprop

Input ranges: [-1 1;-1 1;-1 1;-1 1] Get from inp...

Training function: TRAINLM

Adaption learning function: LEARNGD

Performance function: MSE

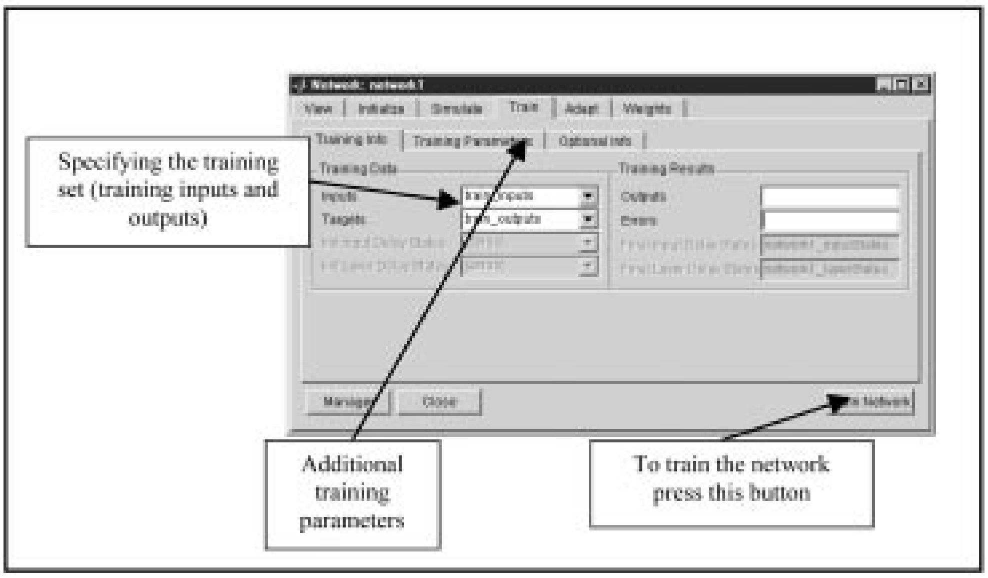
Number of layers: 3

Properties for: Layer 1

Number of neurons: 1

Transfer Function: TANSIG

Fig. 9. (a) Network definition and (b) block-diagram representation of the specific neural network. The number of layers, the neuron transfer functions, the number of inputs, and weight matrices are displayed.


Network: network1

Training Data

Inputs: train_inputs

Targets: train_outputs

Fig. 10. Network training window. The selection of training inputs and targets is intuitive.

window for selecting the training inputs and outputs.

The convergence of the ANN is also displayed visually. Figure 11 shows the mean-squared-error between the simulated and actual outputs as a function of training cycles. Once the performance criterion of a prescribed mean-squared-error is reached, the network solution is ready for simulation with the test data sets. Otherwise, the training is terminated if the maximum training cycle is reached. Again, a window is available to control the simulation tests. The results of the simulation are exported to the MatLab workspace for comparison to the actual outputs. A post-processing algorithm is provided to the students for this final operation. A write-up of the exercise is required along with a screenshot of the error plot [13]. Several networks must be run and documented with a discussion of the network configurations giving the best results.

IMPLEMENTATION

Description

The neural networks module was implemented in an upper-level/introductory graduate course entitled 'Smart Materials and Sensors' [11] during the 2003 Fall semester. The technical interest area was smart structures with an emphasis on intelligent sensing and control technologies. Topics included material properties, stress-strain relationships, strain sensing and networks, electrical resistance gages, linear variable differential transformers, fiber optic sensors, neural networks, and smart bridge applications. The learning objectives of this applications-oriented course were (1) to integrate cross-disciplinary knowledge, (2) to build collaborative skills, and (3) to gain related applied experience. One week was devoted to the topic of neural networks. The two function approximation problems and a VLSI-circuit-feature identification problem were the only examples used. Iris flower identification problem was the only exercise used. The additional examples as given on the website were added based on the assessment.

Sixteen students with interests and background spanning the course topics participated. The average G.P.A. was 3.500/4.00. A majority had not had any previous exposure to neural networks. The student demographics were typical for a course at this level. Ten students were in graduate school and six were undergraduates. Twelve students were male and four were female.

The neural networks component included a ninety-minute lecture, a homework assignment similar to the initial training illustration, the MatLab exercise, an exercise write-up, and a component assessment. Fifty minutes of in-class time was devoted to the MatLab exercise. The students completed the exercise and the write-up out of class. All students found ANN design and training selections that resulted in no error or minor errors in classification for the test data set. However, no student created an ANN with no errors on the first try.

Assessment

The ANN module was assessed through a series of multiple-choice questions and rated statements. The assessment questions and statements concerned the instructional effectiveness of specific components and of the module as a whole. In particular, the lecture, examples, the exercise, and the MatLab environment were addressed. Figure 12 shows the student evaluation of the three components of the module. Eleven students felt that the lecture was sufficient background for the exercise and two felt more and less were needed. All students regarded the lecture examples as beneficial with six students wanting more realistic

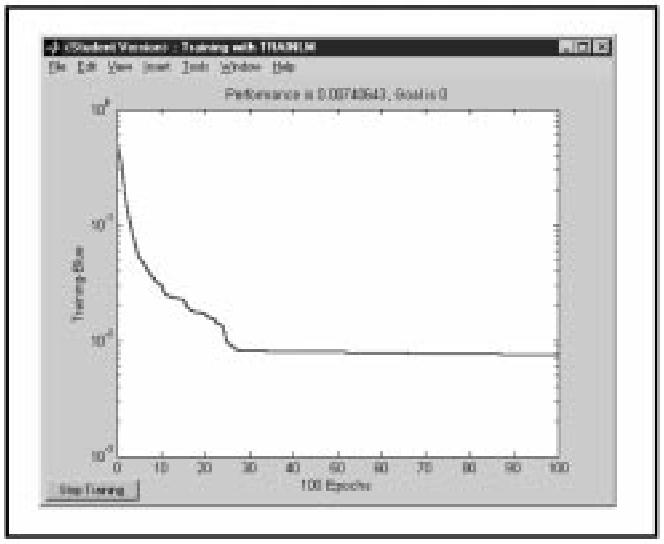

(Student Version) : Training with TRAINLM
File Edit View Insert Tools Window Help
Performance is 0.00740643, Goal is 0
Training-Blue
0 10 20 30 40 50 60 70 80 90 100
100 Epochs
Stop Training

Fig. 11. Variation of mean-squared error as a function of training cycles.

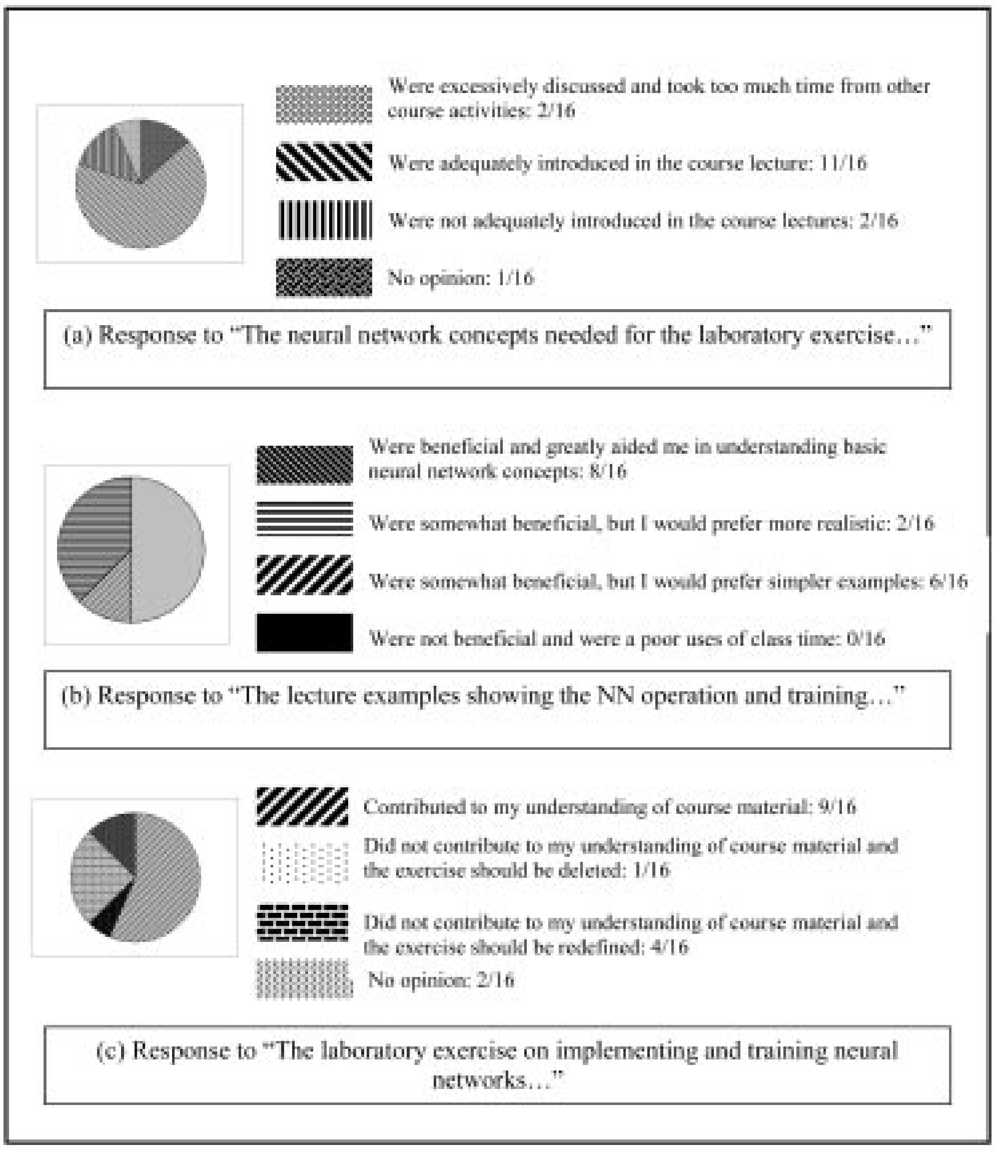


Fig. 12. Responses to multiple-choice questions regarding course components.

cases. A majority of students liked the MatLab iris flower exercise, while a few wanted a redefined exercise. Figure 13 shows results for the overall experience and the MatLab environment. A great majority felt that the module was a valuable introduction and that the MatLab environment facilitated the experience. (The intent of the instruction was not training students to use ANNs, but to understand basic operations. Student responses reflected this limitation.)

The students rated their agreement or disagreement with several statements. In all cases, the scale ranged from '1' for strongly disagree to '10' for strongly agree. Table 1 shows the results. Again, the student response was positive. For the limited scope of the module, the students rated their learning, the content, and the approach as effective. For instance, Question D inquired whether students gained a qualitative/better understanding of the capabilities of neural networks from the lecture. The average response was a 7.440, which signifies that the response is on the positive side. (Note that Questions C and H were asked in such a way that a low average number would be a positive result). Also, most students found the time devoted to the various components adequate. Among the most positive ratings were those for the effectiveness and value of the qualitative instruction in ANNs. The ratings indicated an uncertainty as to whether additional ANN content in this sensors course was desirable. The ratings showed some weakness in the examples and the iris flower exercise. The students seemed to desire more applications that are related to sensors and electrical engineering. Note that GUI-based MatLab was considered a user-friendly environment.

The ANN module was modified to reflect the results of the assessment. The examples and the MatLab exercise were improved to further emphasize applications and to address sensor issues.

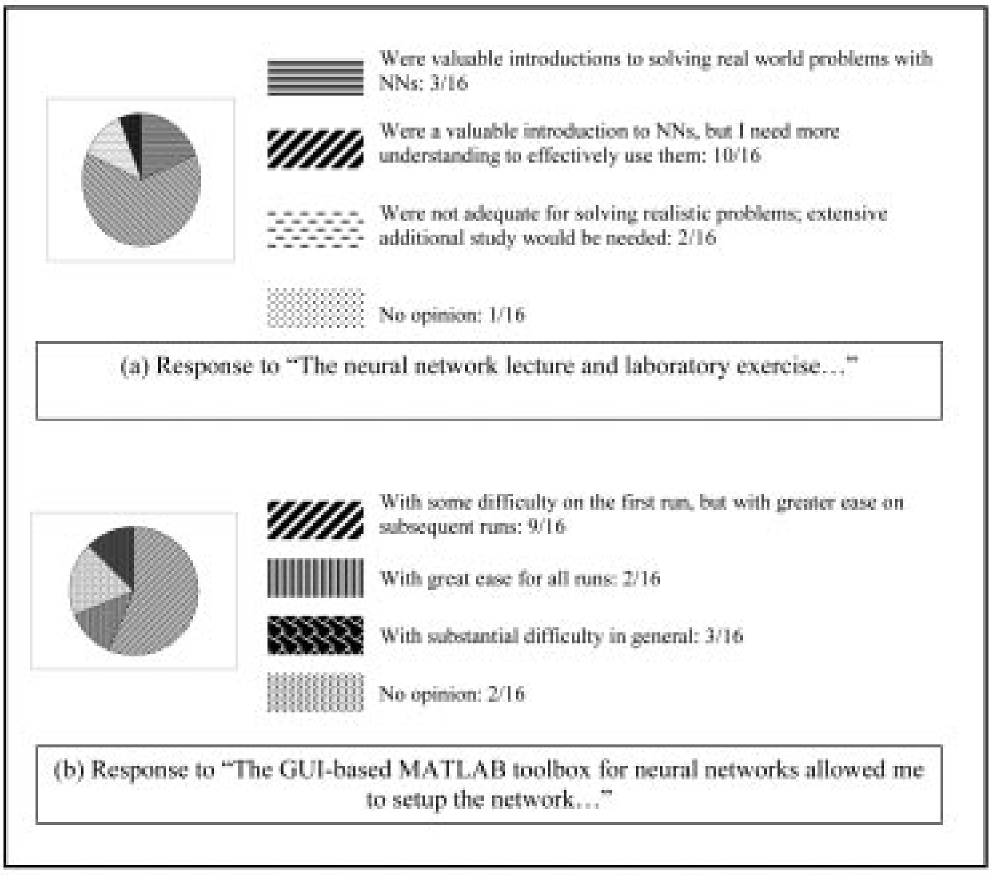


Fig. 13. Responses to multiple-choice questions regarding the module and MatLab.

The resources on the website are the modified version. The lecture was expanded to fill two full fifty-minute lecture periods. The lecture engineering example section was changed and expanded to better relate to the course. Two new ANN problems concerning structural sensing were added and the associated discussion has more detail. The applications are shown using MatLab screenshots to help prepare for the laboratory exercise. Also, additional sensor-related options for the MatLab laboratory are under development to supplement the IRIS-flower identification problem. Many other options are possible for examples and for exercises. Any expansions should relate to the course using the module and the associated ANN should have a limited number of outputs, inputs, and layers for the feedforward back propagation architecture. Additional applications-oriented exercises should be easy to implement, while allowing wide variation in the outcome, i.e. training choices should give networks that produce non-convergent, poorly-convergent, and well-convergent solutions.

CONCLUSION

The instructional module provides a systems-level, applications-oriented introduction to ANNs. It is designed for non-ANN curricula such as structural sensing courses. It addresses the need for non-ANN specialists to collaborate with ANN specialists. The student-centered objectives are to convey awareness of discipline-specific ANN concepts, ANN capabilities and training through examples, and connecting links with engineering applications. The one-week module, which is available on a sensors course website, contains a succinct lecture with examples and an interactive exercise. With a qualitative understanding and a connection to real problems, students are better prepared to collaborate on project teams involving ANNs.

The module was implemented in a smart sensors course and was revised based on an assessment. The overall student response was positive especially regarding the MatLab-based interactive exercise. As per the module design and the limited time, the instruction emphasized ANN capabilities and connecting concepts without mathematical detail and programming experience. The students should be able to interact with ANN specialists and to read ANN literature with general comprehension. Several observations can be drawn. Examples and exercises need to be realistic and to be linked to student interests. Exercises should balance the constraints to limit student design choices to target concepts, with options that will produce a range of ANN performance. The lecture

Table 1. Ratings of the ANN module and experience

Question	Ave.	1	2	3	4	5	6	7	8	9	10
A	7.625	0	1	0	0	3	0	4	1	2	5
B	7.313	0	1	0	0	1	3	5	1	1	4
C	3.000	4	3	4	1	3	0	1	0	0	0
D	7.440	0	0	1	1	3	1	0	4	1	5
E	7.250	0	0	2	0	1	4	2	1	1	5
F	6.625	0	1	1	2	2	2	2	1	1	4
G	6.313	0	0	1	0	4	4	3	3	1	0
H	3.625	4	3	0	2	5	0	1	1	0	0
I	6.930	0	2	0	0	1	3	1	6	1	2
J	6.687	0	1	2	1	1	2	1	4	1	3
K	6.880	1	0	0	0	4	2	2	2	3	2
L	5.562	2	1	0	2	5	0	0	3	2	1

Scale 1–10 '1' strongly disagree an '10' strongly agree
A The length of the Neural Network lecture was sufficient to understand the concept.
B The illustrations and examples in the Neural Network lecture were effective.
C I found the Neural Network information to be difficult to understand.
D I gained a qualitative/better understanding of the capabilities of neural networks from the lecture.
E I gained a qualitative/better understanding of the multi-layer feedforward Neural Network from the lecture.
F I believe that the Neural Networks lecture and laboratory were a valuable component of the course.
G The IRIS-flower classification problem was a good application to test the training algorithms.
H The assigned task for the laboratory Neural Network exercise took excessive time.
I The laboratory Neural Network exercise contributed to my understanding of the Neural Network lecture concepts.
J I was able to implement successful Neural Networks without difficulty.
K The GUI-based MatLab user interface for the laboratory Neural Network exercise was easy to use.
L I would prefer more lectures and laboratory assignments on Neural Networks in the course.

concepts and examples provided the necessary preparation for the laboratory experience

The interactive MatLab exercise provided key educational reinforcement and motivation for ANN concepts. This exercise was a capstone experience in which the foundational knowledge was connected to an engineering context. A relatively transparent learning environment, such as GUI-based MatLab, was needed to make the exercise 'doable' and effective within time limitations. Otherwise, students who are not fluent in programming algorithms and the mathematics of ANNs would have difficulty in completing and comprehending any significant exercise in the limited amount of time.

The approach is appropriate for a wide range of curricula and student interests. While the resources assume familiarity with MatLab (but not the ANN toolbox), the GUI controls are intuitive, such that a brief tutorial would give a working MatLab ability. The lecture and exercise could be easily adjusted for other topical areas. The engineering examples in the lecture could be replaced and the iris flower exercise could be supplemented with other ANN applications. If more time is available, exercises could be made collaborative and involve student presentations [12].

Acknowledgement—This work was partially funded by the Mary K. Finley Missouri Endowment and the National Science Foundation (under CRCD Grant # EEC-9872546 with Mary Poats as technical contact).

REFERENCES

1. S. Haykin, *Neural Networks, A Comprehensive Foundation*, 2nd Edition, Prentice Hall, New Jersey (1999) pp. 156–170.
2. W. B. Spillman Jr., Sensing and Processing for Smart Structures, *Proc. IEEE*, **84**(1) 1996, pp. 68–77.
3. S. E. Watkins , Smart Bridges using Fiber Optic Sensors, *IEEE Instrumentation and Measurement Magazine*, **6**(2) 2003, pp. 25–30.
4. R. Dua, V. M. Eller, K. M. Isaac, S. E. Watkins and D. C. Wunsch, Intelligent strain sensing on a smart composite wing using extrinsic fabry-perot interferometric sensors and neural networks, *INNS-IEEE International Joint Conference on Neural Networks 2003*, 20–24 July 2003, Portland, OR.
5. R. Dua, S. E. Watkins and D. C. Wunsch, Demodulation of extrinsic Fabry-Perot interferometric sensors for vibration testing using neural networks, submitted to *J. Optical Eng.* **43**(12), 2004, pp. 176–187.
6. S. E. Watkins, V. M. Eller, Josh Corra, Martha J. Molander, Bethany Konz, R. H. Hall, K. Chandrashekhara and Abdeldjelil Belarbi, Interdisciplinary Graduate experience: lessons learned, *Proc. 2002 ASEE Conference, 16–19 June 2002*, Montreal, Quebec.
7. L. G. Richards, Promoting Active Learning with Cases and Instructional Modules, *J. Eng. Educ.*, **84**, 1995, pp. 375–381.
8. Hardinge, Competency based instruction: a case study, *Aspects of Educational & Training Technology Series*, **25**, 1992, pp. 289–293.

9. A. B. Buckman, A course in computer-based instrumentation: learning LabView with case studies, *Int. J. Eng. Educ.*, **15**(5) 1999, CD supplement.
10. H. Demuth and M. Beale, *Neural Network Toolbox: For use with MatLab, User's Guide, Version 4*, The MathWorks (2001).
11. S. E. Watkins, R. H. Hall, K. Chandrashekhara and J. M. Baker, Interdisciplinary learning through a connected classroom, *Int. J. Eng. Educ.*, **20**(2) 2004, pp. 176–187.
12. S. E. Watkins and R. H. Hall, Complex problem-solving using structured collaboration, in *Innovations 2003: World Innovations in Engineering Education and Research*, W. Aung, M. H. W. Hoffman, N. W. Jern, R. W. King, and L. M. S. Ruiz, eds, INEER, Arlington, VA (2003) pp. 285–296.
13. S. E. Watkins and R. H. Hall, Educational Resources: Laboratories and documentation on different technologies involved in the development of smart sensors and structures, *Smart Engineering Project*, (2004), available WWW: http://campus.umr.edu/smarteng/.
14. R. Dua, S. E. Watkins, D. C. Wunsch, K. Chandrashekhara, and F. Akhavan, Detection and classification of impact-induced damage in composite plates using neural networks, *INNS-IEEE International Joint Conference on Neural Networks*, (Mount Royal, NJ: International Neural Network Society) (Washington DC, July 2001*)* p. 51.
15. Iris flower description, Brooklyn Botanic Garden, available http://www.bbg.org/gar2/topics/botany/parts_flower_parts.html (2004).

Rohit Dua received his MS in electrical engineering from the University of Missouri–Rolla in 2002 and a BE degree in Electronics and Telecommunication from the University of Pune, Maharashtra, India in 1999. He is currently a Ph.D. candidate at the University of Missouri-Rolla. His research interests include neural networks applied to smart structures and pattern recognition of skin cancer profiles. He is a member of IEEE, the Eta Kappa Nu, and Toastmasters International.

Steve E. Watkins received his Ph.D. in electrical engineering from the University of Texas at Austin in 1989. He is Director of the Applied Optics Laboratory and a Professor of Electrical and Computer Engineering at the University of Missouri–Rolla. He is a 2004 IEEE-USA Congressional Fellow and is a Science and Technology Legislative Aide for Congressman Dana Rohrabacher (46th CA) in Washington, D.C. He has been a visiting physicist at Kirtland Air Force Base and a visiting scholar at NTT in Japan. His research includes smart structures and fiber optic sensor systems. He is a member of SPIE, IEEE, OSA, and ASEE. He is the 2000 recipient of the IEEE Region 5 Outstanding Engineering Educator Award, a 1993 finalist in the Eta Kappa Nu Outstanding Young Engineer Award Program, and a past National Science Foundation Graduate Fellow.

Samuel A. Mulder received his Ph.D. in computer science from the University of Missouri–Rolla in 2004 and a B.S. in computer science from Texas Tech University in 1999. He currently works at Sandia National Laboratory. His research interests include combinatorial optimization, computational intelligence, computer security, and education. He is a member of IEEE and ACM.

Donald C. Wunsch is the Mary K. Finley Missouri Distinguished Professor of Computer Engineering at the University of Missouri–Rolla, where he has been since 1999. His prior positions were Associate Professor and Director of the Applied Computational Intelligence Laboratory at Texas Tech University, Senior Principal Scientist at Boeing, Consultant for Rockwell International, and Technician for International Laser Systems. He received the Ph.D. in Electrical Engineering from the University of Washington, the MS in Applied Mathematics from the same institution, and the B.S. in Applied Mathematics from the University of New Mexico. He has well over 200 publications, and has attracted over $5 million in research funding. He has produced eight Ph.D.'s—four in Electrical Engineering, three in Computer Engineering, and one in Computer Science. Among many awards, he has received the Halliburton Award for Excellence in Teaching and Research, the National Science Foundation CAREER Award, and is an IEEE Fellow. He served as voting member of the IEEE Neural Networks Council, Technical Program Co-Chair for IJCNN 02, General Chair for IJCNN 03, International Neural Networks Society Board of Governors Member, and President-Elect of the International Neural Networks Society.