

## Project 2: *Character Recognition using a Multilayer Feedforward Neural Network and Backpropagation*

---

**Assigned: Thursday, Feb. 9**  
**Due: Sunday, March 5, 2005, 23:59:59 (i.e., before midnight)**

---

### Overview

In this project, you will build a multilayer feedforward neural network using the back propagation algorithm to learn to classify the 10 capital letters A, C, D, E, F, G, H, L, P, and R. A data set from the online UCI (Univ. of California, Irvine) Machine Learning Repository will be used to provide the training, validation, and test data for your neural network. You'll turn in your software that performs the learning, instructions for running your software, and a paper (3-6 pages) that describes your project and results.

### Data Set

The "Artificial Character Database" from the UCI Machine Learning Repository (<http://www.ics.uci.edu/~mlearn/MLRepository.html>) has been downloaded to directory `~parker/courses/cs594ml/Project2`. The original datasets are in the "Original-dataset" subdirectory. However, you DO NOT need to use these original files (although you are welcome to browse through them if you like). The reason is because the original data set represents characters as line segments, rather than pixelated images. For this project, pixelated images are better suited for use as neural network inputs.

So, the original data sets have been converted to grid representations, in which each character is represented as a 12x8 array of pixels, with 0 indicating an "off" pixel and 1 indicating an "on" pixel. This data has been divided into 3 parts, found in the following 3 directories:

- `~parker/courses/cs594ml/Project2/learn-grid/`
- `~parker/courses/cs594ml/Project2/validate-grid/`
- `~parker/courses/cs594ml/Project2/test-grid/`

These three directories have different examples of each of the 10 capital letters your neural net should learn (A, C, D, E, F, G, H, L, P, and R). The "learn-grid/" directory has 100 examples per capital letter, while the "validate-grid/" and the "test-grid/" directories have 250 examples per capital letter. The files are named with the example letter (in lower case), followed by a number (from 1 to 100 (for learn-grid), from 1 to 250 (for validate-grid) and from 251 to 500 (for test-grid)). Example file names are "c39" (which is the 39<sup>th</sup> example of the character "C") and p54 (i.e., the 54<sup>th</sup> example of the character "P"). NOTE: files with the same name, but in different directories are actually different examples. So, "learn-grid/a5" is not the same file as "validate-grid/a5".

The format of each file is 12 rows of 8 digits (either 0 or 1), representing the character example. Additionally, at the end of each file is a 10-element array (indexed as ACDEFGHLPR), with one "1" and nine "0"s, indicating the correct classification of the test character example. As an example, here is the file `learn-grid/c38`:

```
-----  
0 0 0 0 0 1 0 0  
0 0 0 1 1 0 0 0  
0 1 1 0 0 0 0 0  
1 0 0 0 0 0 0 0  
1 0 0 0 0 0 0 0  
1 0 0 0 0 0 0 0  
1 0 0 0 0 0 0 0  
1 0 0 0 0 0 0 0  
0 1 0 0 0 0 0 0  
0 0 1 1 0 0 0 0  
0 0 0 0 1 0 0 0  
0 0 0 0 0 1 0 0  
  
0 1 0 0 0 0 0 0 0 0  
-----
```

If you look closely, you can see a rough “C” in the pattern of 1s in this file. Note that in the last line, the 2nd entry is a 1, corresponding to the correct classification of “C”.

### Design Details

As part of the project, you are responsible for designing your neural network to solve this problem. Example details you must decide include the number of input units, the number of hidden units, learning rate, when to stop training, etc. Be sure you use the correct update rules and output for a multiclass discrimination neural net, as we discussed in class (uses “softmax” function at the output layer, instead of a sigmoid).

### Training the Network

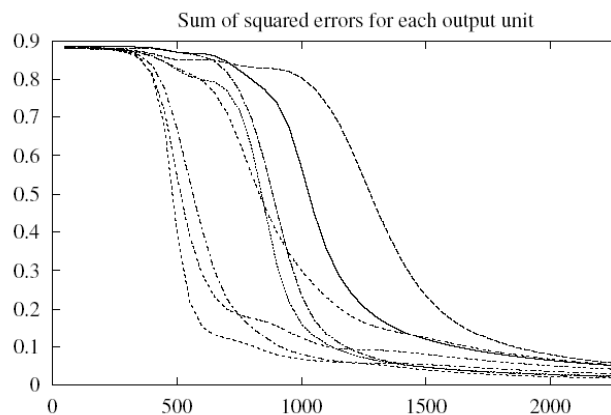
To train your network, you must use the examples in the `learn-grid/` directory. While training the network, you must cross-validate the results using the examples in the `validate-grid/` directory to determine when to stop the training of your network. Keep in mind, that during the validation process, you DO NOT change the weights of your neural network. Your algorithm should go through a process of training on the examples in the `learn-grid/`, validating on the examples in the `validate-grid/` directory, training, validating, training, validating, etc., until the results of the validation (as we discussed in class) indicate to you that the training should stop.

Finally, you use the examples in the `test-grid/` directory to determine the performance of your network (i.e., its accuracy in identifying the characters A, C, D, E, F, G, H, L, P, R) on brand new (never seen) data.

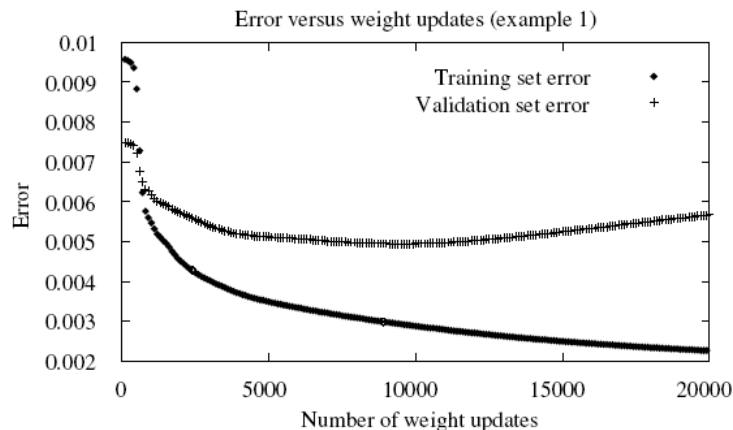
### Data to Gather

To analyze how your network performs, you should gather several types of data and generate plots illustrating this data, which will be included in your report. Specifically, you should collect the following data:

- Sum of squared errors for each output unit (equation 11.14 in your text), as a function of the number of training iterations (i.e., epochs) so far. This data should be plotted on a single graph. An example of this graph is below:



- Error function E (equation 11.18 in your text) for (1) training set, and (2) validation set, as a function of the number of weight updates (i.e., each weight iteration, not each epoch) so far. This data should be plotted on a single graph. An example of this graph is below:



- After your training is completed, you should gather data on the accuracy of your network in identifying characters from the test data set (in the `test-grid/` directory). This data should be presented in a table, giving the success rate for identifying each character, plus the overall success rate for all the test data. (This results in a total of 11 success rates to be reported.)

### **Paper Guidelines**

As part of your project, you must prepare a paper (3-6 pages) describing your project. Your paper should be formatted using common word processing software (such as LaTeX or Word), and should include the following:

- An abstract of 200 to 300 words summarizing your findings.
- An introduction describing the learning task and your formulation of the problem, including the definition of the structure of the neural network and any parameters (such as learning rate) that you used in your system.
- A detailed description of your experiments, with enough information that would enable someone to recreate your experiments.
- An explanation of the results. Use figures, graphs, and tables where appropriate, making sure to include the data mentioned in the previous subsection. Your results should make it clear that the network has in fact learned.
- A discussion of the significance of the results.

### **Undergraduate Grading**

Your grade will be based primarily on the quality of your project implementation and your description of your findings in the paper writeup. You should have a “working” software implementation, meaning that the learning algorithm is implemented in software, it runs without crashing, performs learning iterations as appropriate for a neural network, and is well-documented.

If you have difficulties in getting the network to learn, you will still receive significant credit if you turn in a thorough paper that is readable and clearly outlines your experiments and their results, along with a discussion on why you think you obtained the results you did (or failed to obtain the results you were hoping for). That is, if your neural network does not successfully learn to recognize the specified characters, your work and results should clearly show that you spent considerable time trying to find the correct parameters, network structure, etc. (As we’ve said before, keep in mind that you illustrate this scientifically in the form of research results – figures or graphs that show the results of using different parameters. You *do not* illustrate this by handwaving, making excuses, or saying you tried something for  $x$  hours. Instead, you show quantitative results of those experiments.)

Additionally, you must proofread your paper, ensuring no spelling or grammatical errors (such errors will reduce your grade). Figures and graphs should be clear and readable, with axes labeled and captions that describe what each figure/graph illustrates.

### **Graduate Grading**

Graduate students will be graded more strictly on quality of the research and paper presentation. I expect a more thorough analysis of your results, and a working learning system (i.e., meaning the system actually learns). Your analysis should include a discussion (and perhaps results) on the following points (in addition to the points previously noted above):

- Effect of the number of hidden units on the training time and success rates. (Especially good if you show data illustrating the differences.)
- Effect of different input generalizations (e.g., using a 6x4 input grid instead of the provided 12x8 grid) on the training time and success rates. (Especially good if you give data illustrating the differences.)
- Effect of other system parameters (e.g., learning rate) on the training time and success rates. (Especially good if you give data illustrating the differences.)
- Future work that you believe would improve the learning
- Any other insightful observations you’d like to make

*You should not address these points in a bullet-type fashion, but instead work the answers into your paper in a discussion-style format. The paper should have the “look and feel” of a technical conference paper, with logical flow, good grammar, sound arguments, illustrative figures, etc. Graduate students are expected to format their paper in standard IEEE conference format (see <http://www.ieee.org/portal/pages/pubs/transactions/stylesheets.html> for style files).*

*However, even with this additional information, your paper must not exceed 6 pages.*

### **Turning in your project**

You should email your project to the instructor AND the TA ([parker@cs.utk.edu](mailto:parker@cs.utk.edu); [m Bailey@cs.utk.edu](mailto:m Bailey@cs.utk.edu)) by the deadline.

Your submission should be in 2 parts:

1. Paper (in pdf format)
2. Tar or zip file of the programs and data files needed to run your learning algorithm, including a README file that gives instructions on how to run your code.