Project 1:
Character Recognition using a Multilayer Feedforward Neural Network and Backpropagation

---

Assigned: Tues., September 11

Multiple Due Dates (all submissions must be submitted electronically via Blackboard):

Undergrads: Part 1: Generally complete, compilable code:
due Mon., Sept. 24, 23:59:59 (i.e., midnight)
Part 2: Completed software and paper: due Mon., Oct. 1, 23:59:59 (i.e., midnight)

Grads: Part 1: Completed software: due Fri., Sept. 28, 23:59:59 (i.e., midnight)
Part 2: Completed paper: due Mon., Oct. 1, 23:59:59 (i.e., 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 [for Graduate Students, the paper is 3-6 pages, and describes your project and results; for Undergraduate Students, the paper is 1-3 pages and documents the neural network you implemented and lists the results. More details below].

In this project, you are expected to write all the code yourself, using the programming language of your choice, as long as it can be compiled and run on the UTK EECS linux machines (e.g., hydra). You may not use any canned packages or other resources that have already implemented a neural network for you. The point of this project is to be sure you understand the details of how neural networks are implemented, and to achieve successful learning results using a publicly available data set.

Data Set

The "Artificial Character Database" from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/datasets/Artificial+Characters) has been downloaded to directory `~parker/courses/cs425-528/Project1`. 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/cs425-528/Project1/learn-grid/`
- `~parker/courses/cs425-528/Project1/validate-grid/`
- `~parker/courses/cs425-528/Project1/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 39th example of the character "C") and `p54` (i.e., the 54th 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.

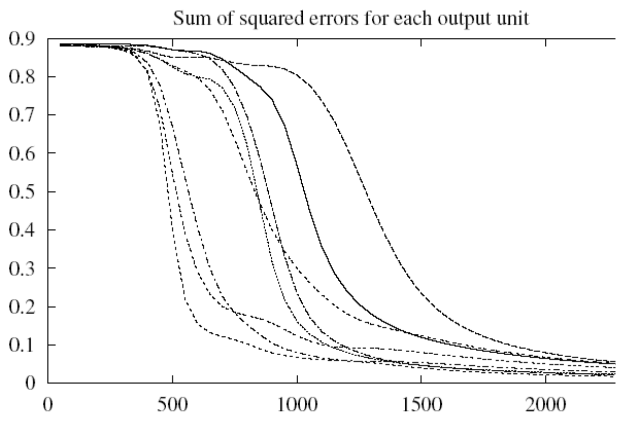
Finally, you use the examples in the `test-grid/` directory to determine the performance of your network (i.e., its accuracy in identifying the characters A, C, D, E, F, G, H, L, P, R) on brand new (never seen) data.

IMPORTANT NOTE: Your code must expect the data files to be in the subdirectories in the ~parker/courses/cs425-528/Project1 directory. Please use only this path in your program, so that your code will run properly when we receive it (i.e., we don't want to have to deal with renaming paths, moving data files around, etc.).

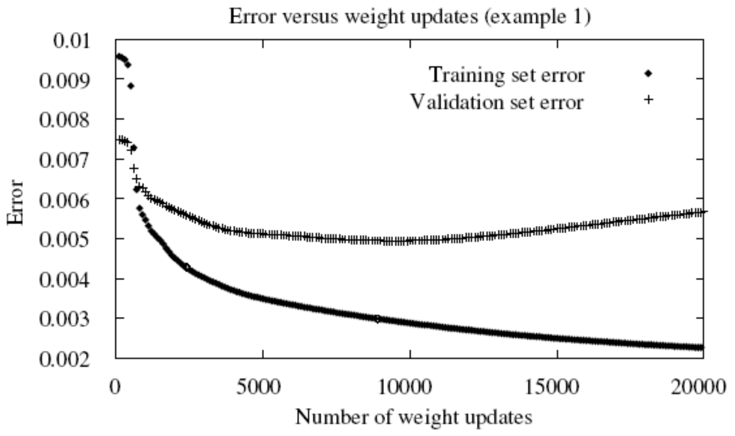
Data to Gather

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 as a function of the number of training epochs (i.e., one complete pass through the training data) so far (equation 20 in my neural net handout). This data should be plotted on a single graph. An example of this graph is below:



- Sum of squared errors for the entire network for a complete epoch as a function of the number of weight updates so far (equation 21 in my neural net handout), for (1) training set, and (2) validation set. 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.)

What your submitted code should do

The final code that you submit should have 2 options – one option is to train the network, and the second option is to test the network on a specific input file. Your code should allow the user to specify whether the neural network should be run in the training mode or the testing mode. This means that your project submission should include the best neural network weights you found during your own training, so that you can use those during testing mode.

It is up to you to design the user interface to allow the user to specify which mode the network should be in. Your user interface should also allow the user to specify a character input file to use for testing your code. These input files will be one of the data files already provided to you in the 3 directories mentioned on page 1 of this document. For example, you might prompt the user to enter the file in the following format: "learn-grid/p54" or "test-grid/c39".

Be sure to provide documentation in your "README" file on how the user should specify the inputs. Or, make your user interface so easy to use that the user just has to follow your program's online instructions.

CS425 – Undergraduate Instructions

Undergraduate Student Paper Guidelines

As part of your project, you must prepare a 1-3 page document (in pdf format) that documents the neural network structure you used and the values of any parameters (such as learning rate) that you used in your system. This document must also include results of your neural network, as described above in the "Data to Gather" section. If you want to present results from several different network configurations, or different parameter settings, that would be terrific. Just be sure to label everything so that it is clear which results correspond to which network structure, parameter settings, etc. To the extent you desire, you are welcome to add additional descriptions along the lines of the graduate student paper. However, this is not required.

The paper you turn in must be in pdf format. However, for this project, you do not have to format it using Latex. (This will come in a later project!) Thus, the format can be anything, as long as it looks professional and you submit it in pdf format. Name your paper *yourlastname*-Project1-Paper.pdf.

Undergraduate Grading

Note that you first must turn in a "generally complete" version of your software, prior to the final version of your software. "Generally complete" means that your code:

- Compiles
- Implements the back propagation algorithm
- Properly inputs and interprets the data files
- Runs without crashing

This code (for the first deadline) does not have to demonstrate good learning performance, however. It just needs to be at the stage that you can tweak the design of the network and the parameters of the system to get it to learn properly. However, it does need to have all the implementation completed for the main input and learning procedures.

For the final submission, you will turn in working software and the paper, as described earlier.

Your grade will be based on:

- 15%: "Generally complete" software (i.e., software with the above characteristics). This software does not have to learn well. But, it has to have all the pieces in place for you to begin tweaking the parameters to achieve good learning performance.
- 65%: The quality of your final working code implementation, and software documentation. 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 learns reasonably well (typically, achieving at least 87% accuracy; it is possible to get as high as 98-99% for some letters). Your final code should have the training mode and testing mode, as outlined earlier in this document.

- 20%: Your paper -- Figures and graphs should be clear and readable, with axes labeled and captions that describe what each figure/graph illustrates. The content should include all the data mentioned previously in this document.

Undergraduates: Turning in your project

You should submit your project via Blackboard by the deadlines. ***Note that you will submit to Blackboard twice, since you have two (2) deadlines for this project.***

Submission #1:

1. This submission is of the "generally complete code". It should include all the programs, include files, makefiles etc., needed to compile and run your project on the EECSCS linux machines (hydra), including a README file that gives instructions on how to compile and run your code. Your main project code file should be called *yourlastname*-Project1-prelim.cc (or whatever extension is appropriate for the programming language you're using). Combine all files together into a single tarball (with extension .tar or .tar.gz or .tar.zip). You DO NOT need to submit the data files we provided you (i.e., the character data set).

Submission #2:

This submission should be in 2 parts:

1. Paper (must be in pdf format, named *yourlastname*-Project1-Paper.pdf)
2. All the programs, data files, include files, makefiles etc., needed to compile and run your project on the EECS linux machines, including a README file that gives instructions on how to compile and run your code. Your main project code file should be called *yourlastname*-Project1.cc (or whatever extension is appropriate for the programming language you're using). Combine all files together into a single tarball (with extension .tar or .tar.gz or .tar.zip). You DO NOT need to submit the data files we provided you (i.e., the character data set).

When we unpack your files, we should be able to read your README file and follow the instructions to compile and run your code. We'll be in a bad mood when grading your work if the instructions you provide do not work ☹, and reserve the right to count off points for egregious cases. So please test your instructions before you send your files to us.

CS528 – Graduate Instructions

Graduate Student Paper Guidelines

As part of your project, you must prepare a paper (3-6 pages) describing your project. Your paper must be formatted using Latex and the standard 2-column IEEE conference paper format. See http://www.ieee.org/web/publications/pubservices/confpub/AuthorTools/conferenceTemplates.html for style files. An example of this paper format is available here: http://web.eecs.utk.edu/~parker/publications/icra10.pdf.

The paper you turn in must be in pdf format. Name your paper *yourlastname*-Project1-Paper.pdf.

Your paper 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 subsections. Your results should make it clear that the network has in fact learned.
- A discussion of the significance of the results.

The reader of your paper should be able to understand what you have done and what your software does without looking at the code itself.

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. Your analysis should include a discussion (and perhaps results) on the following points (in addition to the points previously noted in the paper description 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.)
- Effect of different validation techniques, such as k-fold cross validation (in which case you are free to combine the data from the 3 directories and re-divide into training and validation data as you see fit)
- 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. Be sure to proofread your paper, ensuring no spelling or grammatical errors (such errors will reduce your grade). If you are not a native English speaker, then you are encouraged to employ a fluent English-speaking friend to proof-read your paper to find grammatical and typographical errors. Acknowledge the help of your friend's proof-reading help in an "Acknowledgements" section at the end of your paper. Figures and graphs should be clear and readable, with axes labeled and captions that describe what each figure/graph illustrates.

Your paper must not exceed 6 pages.

Your grade will be based on:

- 65%: Your documented code, which achieves highly accurate results (typically, achieving at least 87% accuracy; it is possible to get as high as 98-99% for some letters). Your final code should have the training mode and testing mode, as outlined earlier in this document.
- 35%: Your paper, which includes an analysis of your neural network project regarding the points mentioned previously

Graduates: Turning in your project

You should submit your project via Blackboard by the deadlines. ***Note that you will submit via Blackboard twice, since you have two (2) deadlines for this project.***

Submission #1:

This submission should include all the programs, include files, makefiles etc., needed to compile and run your project on the EECS linux machines (hydra), including a README file that gives instructions on how to compile and run your code. Your main project code file should be called *yourlastname*-Project1.cc (or whatever extension is appropriate for the programming language you're using). Combine all files together into a single tarball (with extension .tar or .tar.gz or .tar.zip). You DO NOT need to submit the data files we provided you (i.e., the character data set).

When we unpack your files, we should be able to read your README file and follow the instructions to compile and run your code. We'll be in a bad mood when grading your work if the instructions you provide do not work ☹, and reserve the right to count off points for egregious cases. So please test your instructions before you send your files to us.

Submission #2:

This submission is of your paper (in pdf format, named *yourlastname*-Project1-Paper.pdf), formatted in the IEEE conference style format mentioned earlier.