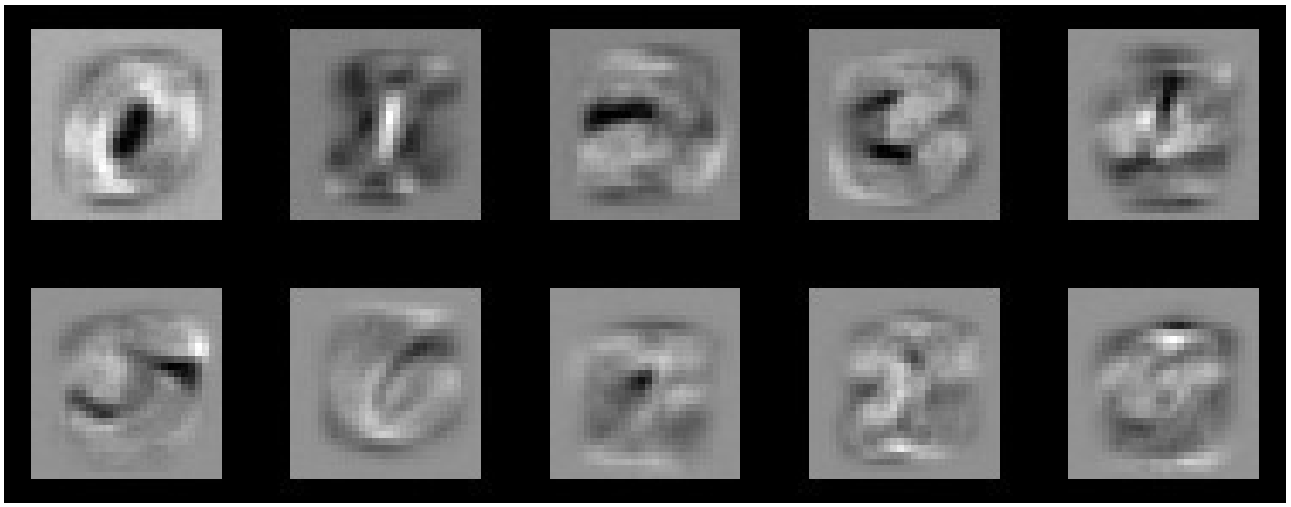
Assignment 4
Neural Networks and Optical Character Recognition

Due date: Friday, Apr. 9
(electronic submission on Quercus)
This assignment can be solve individually or by teams of 2

This assignment is worth 10 units toward the total assigned work you will do during the term



Patterns learned by a neural net
on a handwritten digit classification task

CSC D84 – Assignment 4
Neural Networks – Optical Character Recognition

The goal of this assignment is to let you implement the main components of a neural-network training process. You will experiment with network architecture, try out different activation functions, and become familiar with the information flow over the neural network. You will have to understand how the weights in the network are trained, and once you have a complete implementation, you will be able to observe the network learn in real-time.

Learning Objectives:

You will understand the structure of one and two-layer neural networks. You will observe the layout and connections between neurons, and see the network in action.

You will strengthen your understanding of the information flow within the network. Both in the forward direction (outputs), and in the backward direction (errors).

You will understand how error back-propagation works. You will carry out the back-propagation process in order to adjust the weights linking neurons together.

You will experiment with different network layouts and activation functions, and through this process gain experience about how different neural networks behave.

You will explore a simple (but realistic!) optical character recognition problem that nevertheless shows how more general systems could be built to solve harder problems.

Skills Developed:

Managing neural network structure. Understanding how weights are encoded, stored, and used to compute outputs.

Computing error signals from training data, and using them to adjust the network's weights.

Testing multiple layouts to find one that works best. Comparing the performance of different network architectures.

Reference material:

Your in-class lecture notes on Neural Nets

Digit recognition dataset:

The MNIST database of handwritten digits

http://yann.lecun.com/exdb/mnist/

Optical Character Recognition

Automatic conversion from hand-written text to electronic format is an important application of AI and computer vision. Here you will study a very carefully constrained version of this problem: You will train a Neural Network to recognize hand-written digits such as may appear on mailing envelopes for postal codes, street addresses, and so on.

The specific dataset we will use has been widely studied. It consists of digitized images of digits that have been pre-processed so that the digits all have the same size (28x28 pixels) and a uniform range of brightness values. There is a couple thousand digits, along with associated labels which indicate what digit a particular image represents.

As an OCR problem, this is a reasonably easy one. But the principles you will study in this assignment apply to more general and more difficult OCR problems, as well as to general classification tasks. In particular, Google is currently using Neural Nets for text-to-speech processing on Android devices (http://research.google.com/pubs/SpeechProcessing.html), oh, and to build a machine that can beat humans at Go (https://en.wikipedia.org/wiki/AlphaGo).

While working on this assignment, think about how you would solve a more general OCR problem in which characters have not already been cut and shrunk to a uniform size, and how you would handle variations in handwriting styles (which are much more pronounced for letters than numbers).

Step 1

Download and uncompress the starter code into an empty directory.

This code is designed for Linux. I recommend you do all your work on the CS lab at IC 406. You can work remotely by logging into ***mathlab.utsc.utoronto.ca***.

We will not be able to support Windows, if you choose to develop on Windows you will have to set up your computer properly, and you ***must*** ensure your final submission ***works on mathlab***.

The starter code contains the following files:

NeuralNets_core.o
NeuralNets.[c,h]
REPORT.TXT
Training + Testing data (4 files with .dat extension)

The only program file you need to look at is ***NeuralNets.c***. All your code will be written here.

Read all the comments in NeuralNets.c. The comments describe the global data that is available to you and tell you what needs to be implemented.

Feel free to ask for clarification at any time. But I expect you to have ***read this handout and all the comments in the code carefully***.

Neural Nets for OCR

Step 2

Test-run the starter code:

1) Compile the starter code

2) Run the resulting binary

```
>./NeuralNets 0 0 0 50 0
```

You can run *NeuralNets* without any command line arguments to see a list of parameters. Make sure to have a look before you start working.

If everything works as expected, the program will attempt to train a network and will exit after a short while. Unfortunately, since the neural net training code has not been implemented, the network won't learn anything just yet.

Step 3

Implement the functions required to train a 1-layer neural net and to perform classification with the trained network. The starter code comments will tell you what needs to be implemented and where. Be careful to get the right values for your targets for each neuron, and to compute and propagate output errors correctly.

If you implemented things correctly you should notice that:

* The average classification rates increase after each iteration (quite quickly)
* After some time, they stabilize to the network's maximum classification rate and oscillate around this value.
* The program will then exit and save the network weights and a few images for you to analyze.

If everything works as expected, try running your network in evaluation mode and check that the classification rate is similar to what you obtained in the training set. For a one layer net, you should expect classification rates in the high 80% range.

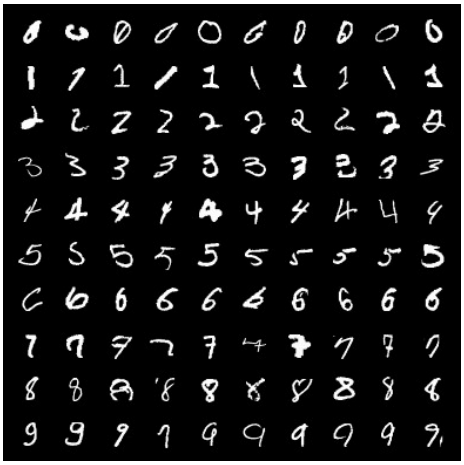
If you run intro trouble:

* Check your calculations and make sure they are sound
* Check for saturation – your neural net's units should not be constantly returning values very close to the minimum or maximum for the function
* If saturation is an issue, check that you are scaling the input appropriately, and if needed, adjust the scaling factor.

Neural Nets for OCR

Step 4

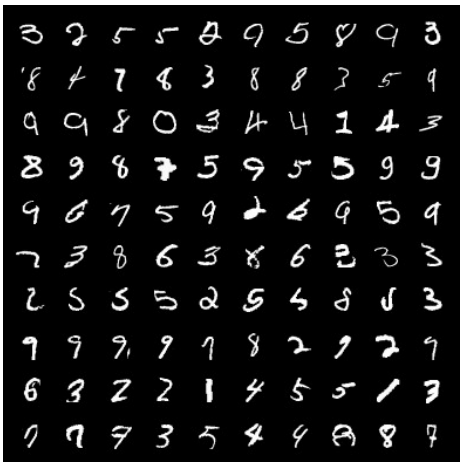
Take a look at the output images and think about what the network is learning/doing.



False Negatives Image

Each row in this image shows you samples of one digit that were incorrectly classified as something else.

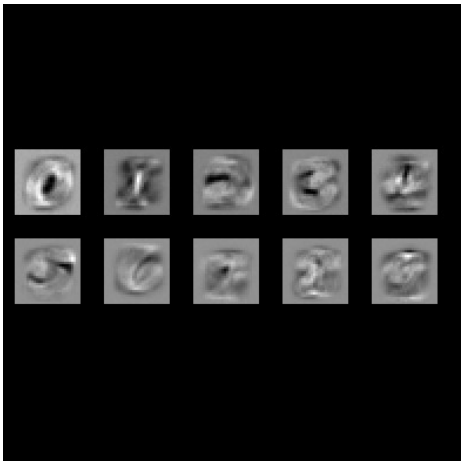
For example, the top row shows samples of ***0*** that were mis-classified as some other digit.



False Positives Image

Each row in this image shows samples of digits that were incorrectly classified as the digit corresponding to the row.

For example, the top row shows examples of digits that were incorrectly classified as ***0***.



Trained Network Weights

This image shows the pattern learned by each of the ouput neurons in terms of weights assigned to input pixels. White is for large, positive weights. Black is for large, negative weights, and gray is close to zero.

** The code can be made to output a movie of the weights changing during training. It's cool to watch, but mind the fact that it requires a good amount of disk space.

Neural Nets for OCR

Step 5

Now complete the code needed to train a 2-layer neural network, and to perform classification Using this network.

This time you have to be particularly careful to make sure you are propagating errors properly from the output layer to and then through the hidden layer. ***Make sure you are not using updated hidden-to-output weights when computing updates for input-to-hidden Weights.***

If all goes well, you should expect to see your network learning, this time you can adjust the Number of hidden-layer neurons. Try different values and see what happens.

If your network is not learning – check all your computations and make sure they are sound, then check for saturation at any layer, and if saturation is occurring, make sure you are scaling the inputs appropriately.

Step 6

Complete all the questions and experiments in REPORT.TXT

Step 7

Submit your work:

Create a single compressed ***.zip*** file – and that means it should actually be ***in .zip format***. If you submit a compressed tar file, a .ar, .arj, .arc, .7z, or anything else renamed to have the .zip extension you will incur the wrath of your TA and lose marks.

The .zip file should contain all your code ***but no images, movies, or digit data***:

```
NeuralNets.[h,c],  REPORT.TXT,  NeuralNets_core.o
```

We are serious about not including additional files. You will be deducted 3 marks for each superfluous File submitted. Name your .zip file:

`NeuralNets_studentNo1_studentNo2.zip` (e.g. NeuralNets_012345678_987654321.zip)

Submit your file on Quercus as usual.

Before you submit, make sure your compressed file in fact contains all your files and code, and that it decompresses properly on mathlab. Non-working .zip files will get zero marks.