# Assignment 4: Neural nets (15%)

Fall 2021 – CPSC 501

Due at 23:59, Dec. 9 on D2L

**Assignment policy:**

- This is an **individual** assignment, so the work you hand in must be your own. Any external sources used must be properly cited (see below).
- Extensions will not be granted to individual students. Requests on behalf of the entire class will only be considered if made more than 24h before the original deadline.
- Some tips to avoid plagiarism in your programming assignments (taken from previous offerings of the course):
  1. Cite all sources of code that you hand in that are not your original work. You can put the citation into comments in your program. For example, if you find and use code found on a web site, include a comment that says, for example:

     `# the following code is from https://www.quackit.com/python/tutorial/python_hello_world.cfm.`

     Use the complete URL so that the marker can check the source.
  2. When you upload your project to GitLab, make sure your repository is set to **private**. Any repository set to public or internal is visible to other students in the class, and counts as academic misconduct.
  3. Citing sources avoids accusations of plagiarism and penalties for academic misconduct. However, you may still get a low grade if you submit code that is not primarily developed by yourself.
  4. Discuss and share ideas with other programmers as much as you like, but make sure that when you write your code that it is your own. A good rule of thumb is to wait 20 minutes after talking with somebody before writing your code. If you exchange code with another student, write code while discussing it with a fellow student, or copy code from another person's console, then this code is not yours.
  5. Collaborative coding is strictly prohibited. Your assignment submission must be strictly your code. Discussing anything beyond assignment requirements and ideas is a strictly forbidden form of collaboration. This includes sharing code, discussing code itself, or modelling code after another student's algorithm. You can not use (even with citation) another student's code.
  6. We will be looking for plagiarism in all code submissions, possibly using automated software designed for the task. Note that this still applies to the current offering.
  7. Remember, if you are having trouble with an assignment, it is always better to go to your TA and/or instructor to get help than it is to plagiarize.

## Instructions:

The goal of this assignment is to give you some practice working with neural nets. The assignment consists of three parts, corresponding to three different data sets. You'll be experimenting with variations of a neural net to solve classification problems, and writing up your results in a report.

For all parts of this assignment, you may use the provided neural network code provided on D2L as network.py. This code is taken from Michael Nielsen's *Neural Networks and Deep Learning* (2015) textbook, which is linked on D2L. The web version is available for free, and is a recommended resource if you want more details on how it works.

All data sets and starter code files are included in the Assignment4.zip file provided on D2L.

**Package management:** It is recommended that you use a python virtual environment to manage any external packages you use for the assignment. Instructions for how to set this up can be found at `https://docs.python.org/3/tutorial/venv.html`. The following packages will be needed for Assignment 4:

- `numpy` – scientific computing library for python (see `https://numpy.org/install/`)
- `matplotlib` – allows easy plotting of data and depiction of images
- `idx2numpy` – useful for reading the MNIST image files

**Part 1: MNIST Dataset:** The home page of the MNIST database is `http://yann.lecun.com/exdb/mnist/`. The data set is divided into a set of 60,000 training examples and a set of 10,000 test examples, each consisting of separate files for the images and their labels. Details on the file format can be found at the bottom of the MNIST web page, and will also be covered in tutorial. Note that Nielsen's textbook trains the net on only 50,000 samples, but you should use all 60,000.

In the starter code, you are provided with hyperparameters for the neural net that will result in roughly 94% accuracy on the test data. Train the neural net with these hyperparameters, and record the results in your report. Next, experiment by adjusting the hyperparameters to achieve an accuracy of 98% or higher on the test data. Record the data for at least two of these neural nets in your report. The neural nets you record should be in order of increasing accuracy, and the final neural net should have accuracy of at least 99%. You may also make changes to improve the speed at which your neural net can be trained, as long as it does not decrease accuracy.

Write a brief paragraph explaining and justifying the changes you made, including relevant code excerpts. You may adjust any of the hyperparameters mentioned. Note that you will need to edit the network.py class to change the activation function if you choose to do so. In your report, it should be clear what changes you made, as well as how/why they improve the net's performance.

In your report for this part, include three images that your final neural net failed to classify correctly. Include the image, the correct label, and the label output by your neural net. You can obtain images by index from the testing data using the imageViewer.py code provided on D2L. The TAs will cover how to do this in tutorial. To do this, it may be useful to save your neural net to a .pkl file after training, and then load it in a separate program to find its output for a particular image.

**Part 2: notMNIST Dataset:** The machine learning community is a bit sick of seeing MNIST digits pop up everywhere, so they created a similar dataset and named it notMNIST. Created by Yaroslav Bulatov, a research engineer previously at Google and now at OpenAI, notMNIST is designed to look like the classic MNIST dataset, but less 'clean' and extremely 'cute'. The images are still 28x28 and there are also 10 labels, representing letters 'A' to 'J'. The homepage for the dataset is `http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html`.

Your task for this dataset is the same as for the MNIST. In this case, however, your starter code hyperparameters will yield an initial accuracy of roughly 89% on the test data. Follow the same instructions as above to reach an accuracy of over 92%.

Write a brief paragraph explaining and justifying the changes you made, including relevant code excerpts. You may adjust any of the hyperparameters mentioned. In your report, it should be clear what changes you made, as well as how/why they improve the net's performance.

In addition to this information, include a short paragraph comparing your results from this dataset to your results from the original MNIST dataset. Note that even though the datasets are similar, it is much harder to get a high accuracy for notMNIST. Why do you think this is the case? Include a brief explanation.

**Part 3: Coronary Heart Disease Dataset:** The file heart.csv contains data taken from `https://web.stanford.edu/~hastie/ElemStatLearn//`. Each entry lists a series of observed health factors, followed by a boolean indicating the presence of coronary heart disease. Your goal is to use the first 9 variables to predict the predict the last. Useful information on the dataset can be found in heartmeta.txt.

You will need to load the data into feature vectors yourself, and divide it into training and testing sets. When converting to feature vectors, you will need to convert family history to a boolean, and rescale the age to have maximum value 1. All other variables should be converted to z-scores using the mean and standard deviation calculated from the dataset. These values are provided in heartmeta.txt so you can check your results, but you should calculate them yourself in your code.

Your task for this dataset is similar to MNIST and notMNIST, but will be more of a challenge. In this case, your starter code hyperparameters will result in the net learning to output a flat value (i.e. nobody gets diagnosed with CHD). Why do you think this is the case? Explain with a sentence or two in your report.

Next, follow the same instructions as for MNIST and notMNIST to find a net that performs better than the initial version. Any accuracy better than a flat value is accepted. One problem you will notice when trying to improve the accuracy for this model is overfitting due to the small number of data samples (although fortunately, this makes the training fast to run).

As in the previous two parts, write a brief paragraph explaining and justifying the changes you made, including relevant code excerpts. You may adjust any of the hyperparameters mentioned. In your report, it should be clear what changes you made, as well as how/why they improve the net's performance.

**Version control:** As in the previous assignments, you will be using either **GitLab or GitHub** to maintain version control and to share your final project with the TAs. Your assignment should be kept in a repository titled CPSC_501_A4. As you develop your code, make sure to use proper version control practices, making regular commits with descriptive messages. This includes **keeping a local copy** of your assignment (including the git repository), on your own computer, just in case :)

**Report:** Create a written PDF report that discusses the variations you tried and their performances. For each neural net you train, you need to keep track of

1. the hyperparameters used to train the net (number of layers, number of neurons in each layer, epochs, step size, batch size, activation function)
2. its performance on training data after each epoch
3. the time required to train the net

Make sure all of these factors are included in your report. Note that you will need to add the timing code yourself to track how long the net.SGD() function takes.

The report should also include **directions for the TA to access your project**. This is how they will be able to access your code and commit history, so double-check this works correctly before submitting. Make sure to indicate in the report whether you decided to implement the bonus part of the assignment. You may also include any information (known bugs, etc.) that you think will be useful to the TAs when grading.

**Submission instructions:**

**Important**: If you are in T05 (Chris' Monday morning tutorial), submit your assignment to **Bardia**, who will be doing your grading. If you are in a different tutorial section, submit to your own TA.

Upload your written report as a **PDF file** to the Assignment 4 dropbox on D2L by 23:59 on December 9th. Make sure you **add the correct TA to your GitLab or GitHub repository with the correct access level** using their email given on D2L. The TA will use the instructions in your report to access your project, through which they will grade your submission.

**Rubric (100 pts total)**:

- Version control: Used Git properly, making multiple small commits with informative messages (5 pts)
- Part 1:
  - Input is correctly handled, including construction of feature vectors and division into training/testing data. (10 pts)
  - At least three neural nets are described, including the one given in the starter code. Description in the report is clear, with appropriate code excerpts. Choice of hyperparameters is well justified and leads to improvements in performance. (10 pts)
  - Final neural net has accuracy at least 99% (5 pts)
  - Three misclassified MNIST images are included, along with both the incorrect outputs and the correct labels from the dataset (5 pts)
- Part 2:
  - Input is correctly handled, including construction of feature vectors and division into training/testing data. (10 pts)
  - At least three neural nets are described, including the one given in the starter code. Description in the report is clear, with appropriate code excerpts. Choice of hyperparameters is well justified and leads to improvements in performance. (10 pts)
  - Final neural net has accuracy at least 92% (5 pts)
  - Comparison is made to MNIST dataset, with reasonable conclusions drawn (5 pts)
- Part 3:
  - Input is correctly handled, including construction of feature vectors and division into training/testing data. (10 pts)

  - At least three neural nets are described, including the one given in the starter code. Description in the report is clear, with appropriate code excerpts. Choice of hyperparameters is well justified and leads to improvements in performance. Report includes observations and solutions to overfitting. (15 pts)
  - Final neural net performs better than flat output (5 pts)
- Logistics: Clear, working instructions on how to access GitLab/GitHub project, submitted as part of the PDF report. (5 pts)